ANALISA BERBANDINGAN KOMPRESI *FILE* PDF DENGAN MENGGUNAKAN ALGORITMA *TABOO CODES* DAN ALGORITMA *LEVENSTEIN*

Nurul Asmidar Simanjuntak[1], Abdul Sani Sembiring[2],
Sumiaty Adelina Hutabarat[3]
[1,2,3]Universitas Budi Darma, Medan
[1]nurulsimanjuntak533@gmail.com, [2]gurkiy@gmail.com,
[3]sumiatyadelina@gmail.com

ABSTRAK

Seiring dengan pesatnya perkembangan ilmu pengetahuan, cara manusia dalam mengakses informasi juga mengalami perubahan signifikan. Kebutuhan informasi kini tidak hanya dipenuhi melalui media cetak, tetapi juga melalui media elektronik seperti *e-book* (buku digital). Untuk mengatasi masalah terkait ukuran besar file digital, seperti file PDF, teknik kompresi atau pemampatan data menjadi sangat penting. Kompresi merupakan proses mengurangi ukuran data menjadi lebih kecil dari ukuran aslinya, sehingga data yang besar dan berisi banyak perulangan karakter menjadi lebih efisien untuk penyimpanan. Penelitian ini bertujuan untuk menilai kinerja berbagai *algoritma* kompresi dengan mengukur hasil kompresi file PDF menggunakan parameter tertentu. Hasil penelitian menunjukkan bahwa *algoritma Levenstein* secara umum lebih efektif dalam menghasilkan ukuran file yang lebih kecil dibandingkan dengan *algoritma Taboo Codes*.

Kata Kunci: kompresi, file PDF, algoritma levenstein, taboo codes

ABSTRACT

With the rapid advancement of scientific knowledge, the way people access information has also changed significantly. The need for information is now met not only through print media but also through electronic media such as e-books. To address issues related to the large size of digital files, such as PDFs, data compression techniques become crucial. Compression is the process of reducing the size of data to be smaller than its original size, making large files with many repeated characters more efficient for storage. This research aims to evaluate the performance of various compression algorithms by measuring the results of compressing PDF files according to specific parameters. The study shows that the Levenstein algorithm generally performs better in producing smaller file sizes compared to the Taboo Codes algorithm.

Keywords: compression, PDF files, levenstein algorithm, taboo codes

A. Pendahuluan

Perkembangan teknologi digital telah secara drastis mengubah cara kita menyimpan, mengirim, dan mendistribusikan informasi dalam bentuk file digital. Kemajuan ini memungkinkan penyimpanan data dalam format yang efisien dan akses yang lebih mudah ke informasi dari berbagai perangkat. Misalnya, file digital seperti dokumen, gambar, dan video dapat disimpan dalam format yang kompresibel untuk menghemat ruang penyimpanan dan memudahkan transfer data (Barkatullah, 2019);(Hildawati, 2024). Teknologi digital juga mendukung berbagai metode distribusi, dari email dan cloud storage hingga platform berbagi file, yang memungkinkan akses cepat dan mudah ke informasi di seluruh dunia (Saputra, A. M. A., Kharisma, L. P. I., 2023). Dengan kemampuan untuk menyimpan dan mentransfer data secara efisien, teknologi digital memfasilitasi pertukaran informasi yang lebih cepat, mendukung kolaborasi global, dan mempercepat inovasi di berbagai bidang, termasuk pendidikan, bisnis, dan hiburan (Yusuf et al., 2020);(Saputra, 2023). Salah satu format file digital yang paling umum digunakan adalah PDF *(Portable Document Format).* Diperkenalkan oleh *Adobe Systems* pada tahun 1993, PDF dirancang untuk memfasilitasi pertukaran dokumen digital dengan mempertahankan format dan tampilan yang konsisten di berbagai perangkat dan platform (Saleh, 2014);(Tanwir, 2023);(Suprihadi, 2020). Format ini banyak diterima dan digunakan dalam berbagai sektor, termasuk pendidikan, bisnis, dan administrasi, karena kemampuannya untuk menyimpan berbagai jenis konten, dari teks dan gambar hingga grafik dan formulir interaktif, serta memastikan dokumen dapat diakses dan dibaca dengan cara yang seragam oleh pengguna di seluruh dunia (Purnama, 2022).

Kebutuhan untuk mentransmisikan file PDF seringkali menghadapi tantangan karena ukuran file yang besar dapat membebani memori penyimpanan dan memperlambat proses transfer data. Ukuran besar ini juga menyebabkan pemborosan ruang penyimpanan, yang dapat mengakibatkan penghapusan data yang dianggap tidak penting (Pratiwi & Nasution, 2018). Untuk mengatasi

masalah ini, teknik kompresi diperlukan untuk mengurangi ukuran file. Penelitian ini menggunakan *algoritma Taboo Codes dan Levenstein* untuk kompresi, dengan tujuan untuk menentukan algoritma yang paling efektif. Kedua algoritma ini diuji dan dibandingkan berdasarkan beberapa variabel, seperti *Ratio Compression (RC), Compression Ratio (CR), Redundancy (RD), dan Space Saving (SS)*, untuk menilai efektivitas masing-masing dalam mengompresi file PDF.

Algoritma Taboo Codes berfokus pada menghasilkan kode optimal dengan mempertimbangkan pembatasan tertentu untuk menghindari kemungkinan jalan buntu. Prinsip dasar algoritma ini melibatkan pemilihan bilangan positif \(n \) dan membalikkan pola \(n \) untuk menandai akhir dari kode. Sebaliknya, *algoritma Levenstein* adalah metode kompresi *lossless* yang memastikan file yang didekompresi identik dengan file aslinya tanpa kehilangan data (Rhamadani, 2022). Algoritma ini mengikuti langkah-langkah yang telah ditentukan selama proses pengkodean dan penyandian.

Dalam kutipan “jurnal Ledi Varia Simanjuntak, dkk (Simanjuntak, 2020)”. Dalam penelitian berjudul "Perbandingan Algoritma Elias Delta Code Dengan Levenstein Untuk Kompresi File Teks," ditemukan bahwa baik algoritma Elias Delta Code maupun Levenstein efektif dalam mengompresi file dengan ekstensi *.txt. Program yang dikembangkan menggunakan Visual Basic .NET 2008 menunjukkan kinerja yang baik dalam proses kompresi dan dekompresi file. Hasil penelitian juga mengindikasikan bahwa rasio kompresi meningkat seiring dengan banyaknya pengulangan karakter dalam file yang dikompresi. Berdasarkan penelitian yang dilakukan oleh Arianti Rhamadani, dkk (Rhamadani, 2023) pada tahun 2022 Dalam analisis perbandingan antara *Algoritma Taboo Codes* dan *Algoritma Yamamoto’s Recursive Code* untuk mengompresi file teks menggunakan metode exponential, dapat disimpulkan bahwa kedua algoritma efektif dalam kompresi dan dekompresi file teks. Algoritma Taboo Codes menghasilkan space saving sebesar 36,25%, sedangkan Algoritma Yamamoto’s Recursive

Code mencapai space saving sebesar 25%. Namun, berdasarkan metode perbandingan exponential, *Algoritma Yamamoto's Recursive Code* terbukti lebih efektif dan efisien dalam melakukan kompresi file teks. Dan pada penelitian sebelumnya dengan "jurnal Bobby Ramadhana, dkk (Ramadhana, 2021)". Dengan judul "Implementasi Kombinasi Algoritma *Fibonacci Codes* Dan *Levenstein Codes* Untuk Kompresi *File* PDF". Disimpulkan bahwa dari proses kompresi *file* PDF berhasil digunakan di kedua algoritma dan mengalami perubahan dengan rata-rata *space saving* yang didapat lebih dari 50% dan waktu proses pengkompresian bervariasi kerna dipengaruhi oleh panjangnya karakter *file* yang diinputkan.

Berdasarkan latar belakang masalah, penulis berencana untuk mengompresi file PDF menggunakan *algoritma Taboo Codes dan Levenstein*, serta menganalisis perbandingan rasio kompresi dan waktu kompresi dari kedua algoritma tersebut. Tujuan dari studi ini adalah untuk memberikan wawasan mengenai algoritma mana yang lebih baik di antara keduanya. Oleh karena itu, penulis memilih judul penelitian "Analisa Perbandingan Kompresi File PDF Dengan Menggunakan Algoritma *Taboo Codes Dan Algoritma Levenstein*" .

B. Metode Penelitian

Kerangka penelitian ini menjelaskan langkah-langkah sistematis yang akan diambil dalam proses pencarian. Langkah pertama adalah identifikasi masalah, yang bertujuan untuk memahami dan menguraikan masalah utama. Selanjutnya, pada studi masalah, dilakukan pemahaman mendalam terhadap objek penelitian melalui berbagai referensi seperti jurnal dan buku. Penerapan algoritma Taboo Codes dan Levenstein dilakukan untuk kompresi data lossless, di mana data yang dikompresi dapat didekompresi kembali dengan hasil yang identik dengan data asli. Analisa perbandingan algoritma kemudian dilakukan untuk mengevaluasi efektivitas kedua algoritma dalam mengompresi file PDF. Perancangan sistem bertujuan untuk mengimplementasikan algoritma dan menjelaskan alur kerja sistem yang dibangun. Pengujian dilakukan untuk memastikan hasil kompresi file PDF sesuai dengan harapan.

Implementasi melibatkan pembangunan perangkat lunak untuk memverifikasi apakah sistem berfungsi dengan baik atau memerlukan perbaikan. Akhirnya, dokumentasi merupakan tahap penutup yang meliputi laporan hasil penelitian, termasuk proses decoding file terkompresi untuk memastikan ukuran output sesuai dengan file asli atau terkompresi.

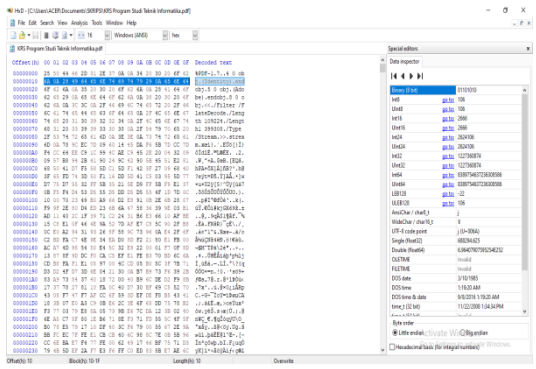
Sampel Data

Dalam penelitian ini data yang diambil yaitu sampel data *file* PDF. Data dapat berupa situasi gambar, huruf, angka, *file* dan dapat digunakan sebagai sumber untuk melihat konsep atau objek. Sampel datanya sebagai berikut.

Tabel 1 Sampel Data

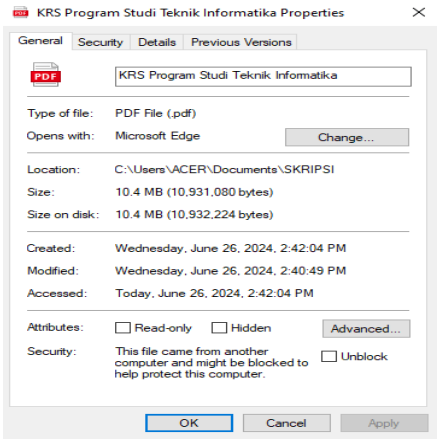
Nama *File*	KRS Program Studi Teknik Informatika
Size	10,4 MB
Type	PDF

Pada penelitian ini digunakan sampel data pada poin pertama, *properties* dari *file* yang akan dijadikan sampel data dapat dilihat pada gambar dibawah ini.



Gambar 1 *Properties* Sampel Data

Berdasarkan tabel diatas maka *file* PDF dapat diubah menjadi nilai *hexadesimal* menggunakan aplikasi HxD.



Gambar 2 Isi *File* PDF Dari Hasil HxD

C. Hasil Penelitian dan Pembahasan

Penerapan Algoritma *Taboo Codes*

Dalam penelitian ini, dua tahap utama dalam pengkompresian teks yang akan dianalisis, yakni proses kompresi dan dekompresi. Penulis akan menjalankan proses kompresi teks menggunakan algoritma *Taboo*

Codes, yang merupakan salah satu teknik kompresi *lossless.*

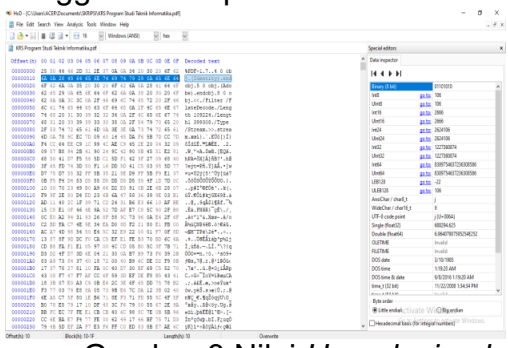
Analisa Proses Kompresi *File* PDF

Analisis proses kompresi *file* PDF dengan algoritma *Taboo Codes*. Berikut merupakan sampel PDF yang akan dilakukan proses kompresi.

Tabel 1 sampel *file* PDF yang akan dikompresi

Nama *File*	KRS Program Studi Teknik Informatika
Size	10,4 MB
Type	PDF

Berdasarkan tabel diatas maka *file* PDF tersebut dapat diubah menjadi nilai *hexadecimal* dengan menggunakan aplikasi HxD.



Gambar 3 Nilai *Hexadesimal* sampel *file* PDF

Mencari Frekuensi

Berdasarkan gambar diatas penulis mengambil 16 sampel data awal yaitu:

Tabel 2Tampilan Nilai Awal Berdasarkan *Hexadesimal*

6A	0A	28	49
64	65	6E	74
69	74	79	29
0A	65	6E	64

Kemudian mengurutkan setiap karakter dalam tabel dimulai dari jumlah frekuensi kemunculan terbanyak jika terdapat jumlah frekuensi kemunculan yang sama maka diurutkan sesuai karakter.

Tabel 3 Nilai Hexadesimal Sebelum Dikompresi

No	Nilai Hexadecimal	Nilai Biner	Bit	Freq	Bit X Freq
1	0A	00001010	8	2	16
2	64	01100100	8	2	16
3	65	01100101	8	2	16
4	6E	01101110	8	2	16
5	74	01110100	8	2	16
6	6A	01101010	8	1	8
7	28	00101000	8	1	8
8	49	01001001	8	1	8
9	69	01101001	8	1	8
10	79	01111001	8	1	8
11	29	00101001	8	1	8
Total					128

Setelah bilangan *hexadecimal* diurutkan berdasarkan frekuensi kemunculan dan didapatkan nilai *biner*, maka selanjutnya menghitung *bit* dan pengurutan *code* algoritma *Taboo Codes* serta memperoleh *bit file* terkompresi.

Langkah selanjutnya menambahkan *codeword Taboo Codes*

Tabel 4 Kompresi Nilai *File* PDF Sampel Dengan Algoritma *Taboo Codes*

No	Nilai Hexadesimal	Nilai Biner	Bit	Freq	Bit X Freq
0	0A	0100	4	2	8
1	64	1000	4	2	8
2	65	1100	4	2	8
3	6E	010100	6	2	12
4	74	011000	6	2	12
5	6A	011100	6	1	6
6	28	100100	6	1	6
7	49	101000	6	1	6
8	69	101100	6	1	6
9	79	110100	6	1	6
10	29	111000	6	1	6
Total					84

Masukkan nilai hexadecimal sampel

Berdasarkan gambar 4. nilai *hexadesimal* diatas penulis mengambil 16 sampel data awal yaitu:

Tabel 5 nilai *hexadesimal*

6A	0A	28	49
64	65	6E	74
69	74	79	29
0A	65	6E	64

Melakukan pembacaan isi *file*

Kemudian mengurutkan setiap karakter dalam tabel dimulai dari jumlah frekuensi kemunculan yag sama maka diurutkan sesuai karakter.

Tabel 6 Nilai Hexadesimal sebelum dikompresi

No	Nilai Hexadesimal	Nilai Biner	Bit	Freq	Bit X Freq
1	0A	00001010	8	2	16
2	64	01100100	8	2	16
3	65	01100101	8	2	16
4	6E	01101110	8	2	16
5	74	01110100	8	2	16
6	6A	01101010	8	1	8
7	28	00101000	8	1	8
8	49	01001001	8	1	8
9	69	01101001	8	1	8
10	79	01111001	8	1	8
11	29	00101001	8	1	8
Total					128

Letakkan tabel codelevenstein

Setelah bilangan *Hexadesimal* diurutkan berdasarkan frekuensi kemunculan dan didapatkan nilai biner, maka selanjutnya menghitung *bit* dan pengurutan *code* algoritma *Levenstein* serta memperoleh *bit file* terkompres, adapun proses *file* PDF dapat dilihat pada tabel dibawah ini.

T

abel 6 Nilai Hexadesimal setelah dikompresi

No	Nilai Hexadesimal	*Codeword*	Bit	Freq	Bit X Freq
1	0A	10	2	2	4
2	64	1100	4	2	8
3	65	1101	4	2	8
4	6E	1110000	7	2	14
5	74	1110001	7	2	14
6	6A	1110010	7	1	7
7	28	1110011	7	1	7
8	49	11101000	8	1	8
9	69	11101001	8	1	8
10	79	11101010	8	1	8
11	29	11101011	8	1	8
Total					94

Berdasarkan data pada tabel diatas dapat dibentuk dalam nilai bit hasil kompresi dari susunan nilai *hexadesimal* pada sampel awal sebelum kompresi yaitu:

6A 1110010	**0A** 10	**28** 1110011	**49** 11101000
64 1100	**65** 1101	**6E** 1110000	**74** 1110001
69 11101001	**74** 1110001	**79** 11101010	**29** 11101011
0A 10	**65** 1101	**6E** 1110000	**64** 1100

Kemudian, dari hasil jumlah *bit* x frekuensi didapat nilai 94, perlu ditambahkan jumlah *bit* kerena 94 tidak habis dibagi dengan 8 dalam konteks komputer, setiap karakter yang menggunakan *ASCII(American Standard Code for Information Interchange)*, yang terdiri dari 8 *bit biner*. Apabila jumlah *bit* data lebih pendek dari 8 *bit,* biasanya komputer tidak dapat membaca secara langsung sebelum proses enkripsi dilakukan. Selama proses enkripsi*, bit* data akan dikodekan secara berurutan setiap 8 *bit* untuk membentuk karakter yang dapat dibaca oleh komputer. Maka dibentuk dua variable yang disebut dengan *padding* dan *flagging* untuk menambahkan *bit* data. *Padding* adalah penambahan *bit* data yang telah dikompresi sehingga jumlah total *bit* data tersebut menjadi kelipatan 8. Sementara *flagging* adalah menambahan 8 *bit* bilangan *biner* setelah *padding* yang digunakan untuk memudahkan pembacaan *bit-bit* hasil kompresi selama proses dekompresi.

Tabel 7 Penambahan *Padding* Dan *flagging* Dengan Algoritma *Levenstein*

Padding	*Flagging*
94 mod 8 = 6 = n 7 - 6 + ”1” 7 – 6 + ”1” = 01	9 - n 9 - 6 = 3 = 00000011

11100101011100111110100011001
11011110000111000111101001111
10001111010101110101110110111
11000011000100000011

Gambar 8 Penambahan *Padding* Dan *Flagging Bit* Pada *String Bit*

Total panjang keseluruhan *bit* setelah melakukan *padding* dan *flagging bit* adalah 94+10 = 104 *bit*. Lalu lakukan pengelompokan bit kebeberapa kelompok. perkelompok terdiri atas 8 *bit* seperti gambar dibawah ini:

11100101 01110011 11101000
11001101 11100001 11000111
10100111 10001111 01010111
01011101 10111100 00110001
00000011

Gambar 9 Pembagian *String Bit*

Berdasarkan hasil pemecahan *bit* diatas, terbentuk 13 kelompok dengan nilai biner terkompresi baru serta nilai biner tambahan. Kemudian langkah selanjutnya adalah mengubah nilai biner ke nilai hexadesimal untuk menemukan karakter yang dihasilkan yang cocok dengan kode ASCII untuk mengetahui nilai yang sudah dikompresi.

Tabel 8 Hasil Hexadesimal Terkompresi

No	Biner	Decimal	Hexadesimal	Karakter
1	11100101	229	E5	å
2	01110011	115	73	s
3	11101000	232	E8	è
4	11001101	205	CD	Í
5	11100001	225	E1	á
6	11000111	199	C7	Ç
7	10100111	167	A7	§
8	10001111	143	8F	□
9	01010111	87	57	W
10	01011101	93	39	]
11	10111100	188	BC	¼
12	00110001	49	31	1
13	00000011	3	3	

Analisa Perbandingan Algoritma *Taboo Codes* Dan Algoritma *Levenstein*

Analisa perbandingan antara dua algoritma memerlukan langkah-langkah sistematis untuk mengevaluasi performa masing-masing. Pertama, kriteria perbandingan ditentukan, meliputi Ratio Compression (CR), Compression Ratio (CR), Redundancy (RD), dan Space Saving (SS). Selanjutnya, nilai untuk setiap kriteria dilaporkan berdasarkan analisis yang dilakukan pada algoritma Taboo Codes dan Levenstein dalam kompresi file PDF. Akhirnya, hasil perbandingan dikumpulkan dan dievaluasi untuk menentukan prioritas atau

keunggulan salah satu algoritma berdasarkan nilai yang diperoleh dari kriteria tersebut. Hasil keputusan ini biasanya disajikan dalam tabel untuk memudahkan interpretasi dan perbandingan performa antara algoritma.

Tabel 9 Prioritas Keputusan

Algoritma	*RC*	*CR*	*RD*	*SS*	*Rank*
Algoritma *Taboo Codes*	1,523	65,625%	34,375%	34,375%	2
Algoritma *Levenstein*	1,36	73,437%	26,563%	26,563%	1

Dari tabel 4.17 diatas dapat disimpulkan bahwa algoritma *Levenstein* lebih efisien dalam hal *RC,CR,RD,SS* dibandingkan dengan algoritma *Taboo Codes*. Dikerenakan algoritma yang memiliki nilai *RC,CR,RD,SS* paling tinggi maka algoritma tersebut lebih baik digunakan dalam kompresi *file* PDF.

Tampilan Sistem

Tampilan sistem merupakan sebuah perwujudan dari rancangan aplikasi yang telah di sketsa sebelumnya. Didalam sistem aplikasi ini akan terbentuk tampilan yang terdiri atas *Form* Menu Utama, *Form* Kompresi Dan *Form* Dekompresi.

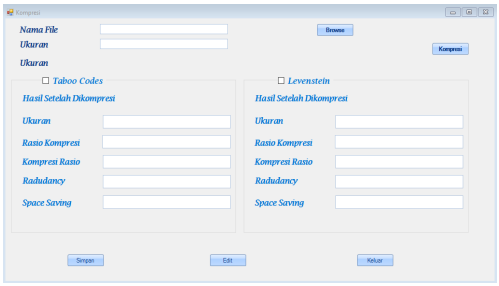
1. *Form* Menu Utama

Form Menu Utama menjalankan sistem kompresi untuk *file* PDF, formulir menu utama muncul terlebih dahulu. tampilan ini memiliki beberapa tampilan untuk mengakses *form-form* yang terdapat dalam sistem. Berikut *form* menu utama:

Gambar 4 *Form* Menu Utama

2. *Form* Kompresi

Form kompresi adalah tampilan *form* yang memungkinkan sistem untuk mengompresi *file* PDF yang diinput. Ketika *file* PDF dikompresi, dapat membuat *file* PDF menjadi ukuran yang lebih kecil.

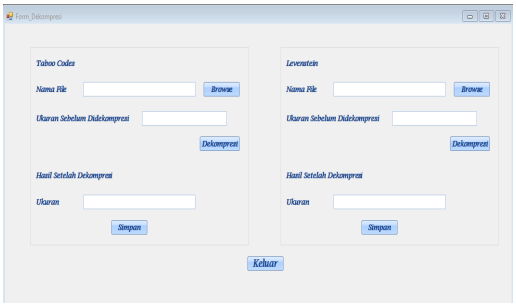


Gambar 5 *Form* Kompresi

3. *Form* Dekompresi

Form Dekompresi adalah tampilan *form* yang memungkinkan sistem melakukan proses dekompresi *file* PDF yang dimasukkan. Setelah *file* PDF terkompresi didekompresi, *file*

PDF asli yang tidak dilakukan kompresi.



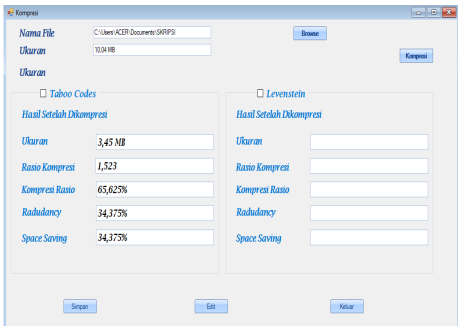
Gambar 6 *Form* Dekompresi

Pengujian Sistem

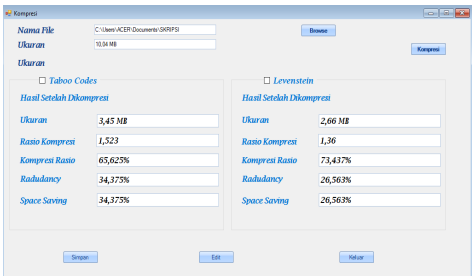
Tujuan dari pengujian sistem yaitu untuk mengetahui begaimana sistem yang dibangun bejalan. Sistem dirancang sesederhana mungkin sehingga pengguna dapat dengan mudah mengaksesnya. Dalam hal ini, penulis memberikan *printout* (hasil cetakan) saat sistem sedang berjalan.

1. *Form* Kompresi

   Form ini menunjukkan proses kompresi *file* PDF menggunakan algoritma *Taboo Codes* dan algoritma *Levenstein* yang *user* input terlebih dahulu *file* PDF yang akan dikompres. Selanjutnya memilih algoritma mana yang akan digunakan dalam pengkompresian *file* PDF 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 7 tampilan *form* kompresi pada saat menampilkan hasil kompresi



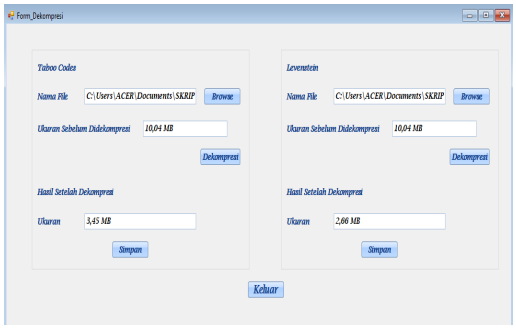
Gambar 8 Menampilkan Hasil Kompresi Dari Kedua Algoritma

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

2. *Form* Dekompresi

   Form dekompresi akan digunakan dalam proses dekompresi *file* atau

mengembalikan *file* PDF kedalam ukuran semula, adapun cara untuk mengakses *form* dekompresi yaitu dengan mengklik menu dekompresi maka halaman dekompresi akan terbuka. kemudian pilih *file* PDF yang akan didekompresi dengan cara mengklik *brwose* maka akan muncul *file* PDF yang telah dikompresi.



Gambar 9 Proses Dekompresi *file* terkompresi

Setelah *file* PDF terkompresi telah berhasil didekompresi, yaitu *file* PDF yang didekompresi berukuran sama dengan *file* aslinya. Pengguna kemudian dapat memilih *menu* simpan dan tombil keluar.

Hasil Pengujian Sampel

Perbedaan antara *file* PDF sebelum kompresi dan setelah kompresi dapat dilihat dari hasil pengujian di bawah ini:

Tabel 13 Hasil Penguji Kompresi

Nama *File*	**Ukuran data Sebelum dikompresi**	**Ukuran Data Sesudah Dikompresi**	
		Taboo Codes	***Levenstein***
KRS Program Studi Teknik Informatika	10.04 MB	3.45 MB	2.66 MB

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

D. Kesimpulan

Berdasarkan hasil implementasi dan pengujian aplikasi HxD untuk analisis sistem, kesimpulan yang dapat diambil dari uji coba kompresi dan dekompresi file PDF dengan algoritma Taboo Codes dan Levenstein adalah sebagai berikut:

pertama, kedua algoritma, Taboo Codes dan Levenstein, dapat diimplementasikan untuk kompresi file PDF. Kedua, hasil penerapan menunjukkan bahwa Algoritma Taboo Codes memiliki nilai Ratio of Compression 1,523, Compression Ratio 65,625%, dan Space Saving 34,375%, sedangkan Algoritma Levenstein memiliki nilai Ratio of Compression 1,36, Compression Ratio 73,437%, dan Space Saving 26,563%. Terakhir, aplikasi yang diusulkan dapat mengompresi file PDF menggunakan kedua algoritma ini dan melakukan perbandingan hasilnya dengan Microsoft Visual Studio 2010, serta berfungsi dengan sangat baik.

DAFTAR PUSTAKA

Barkatullah. (2019). Hukum Transaksi Elektronik di Indonesia: sebagai pedoman dalam menghadapi era digital Bisnis e-commerce di Indonesia. In *Nusamedia.*

Hildawati. (2024). Literasi Digital: Membangun Wawasan Cerdas dalam Era Digital terkini . In *PT. Green Pustaka Indonesia.*

Pratiwi, R. D., & Nasution, S. D. (2018). PERANCANGAN APLIKASI KOMPRESI FILE TEKS DENGAN MENERAPKAN ALGORTIMA FIXED LENGTH BINARY ENCODING (FLBE). *MEDIA INFORMATIKA BUDIDARMA*, *2*(1), 10–14.

Purnama, A. N. (2022). Implementasi Algoritma Rice Code dan Ternary Comma Code Pada Kompresi File PDF. *KOMIK (Konferensi Nasional Teknologi Informasi Dan Komputer)*, *6*(1), 34–42. https://doi.org/10.30865/komik.v6i1.5787

Ramadhana, B. (2021). *Implementasi Kombinasi Algoritma Fibonacci Codes Dan Levenstein Codes Untuk Kompresi File Pdf*. *8*(2), 67–71.

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.

Rhamadani, A. (2023). 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*(1), 140–150. https://doi.org/10.30865/komik.v6i1.5728

Saleh. (2014). Pengembangan perpustakaan digital. In *Tangerang Selatan: Universitas Terbuka.*

Saputra, A. M. A., Kharisma, L. P. I., R. (2023). KNOLOGI INFORMASI: Peranan TI dalam

berbagai bidang . In *PT. Sonpedia Publishing Indonesia.*

Saputra. (2023). TEKNOLOGI INFORMASI: Peranan TI dalam berbagai bidang . In *PT. Sonpedia Publishing Indonesia.*

Simanjuntak, L. V. (2020). Perbandingan Algoritma Elias Delta Code dengan Levenstein Untuk Kompresi File Teks. *Journal of Computer System and …*, *1*(3), 184–190.

Suprihadi. (2020). Sistem informasi bisnis dunia versi 4.0. In *Penerbit Andi.*

Tanwir. (2023). Media pembelajaran berbasis digital. In *(Teori & praktek).*

Yusuf, M. R., Zulaikha, S. R., & Pendahuluan, A. (2020). Perkembangan pengelolaan arsip di era teknologi. *ACARYA PUSTAKA*, *7*(1).