Jurnal Sains Dan Teknologi Informasi
Vol 4, No 1, Desember 2024, Hal 39-45
ISSN 2809-610X (Media Online)
DOI 10.47065/jussi.v4i1.7706
Website https://ejurnal.seminar-id.com/index.php/jussi

Perancangan Aplikasi Kompresi File PDF Dengan Menerapkan Algoritma Stout Code

Indra Wardi

[1] Fakultas Ilmu Komputer dan Teknologi Informasi, Program Studi Teknik Informatika, Universitas Budi Darma, Medan, Indonesia
Jalan Sisingamanganraja No. 338, Medan, Sumatera Utara, Indonesia
Email: indranazara94@gmail.com
(*: Coressponding Author)

Abstrak−Kebiasaan manusia dalam mengumpulkan dan bertukar data salah satunya berupa file PDF memberikan dampak akan terbatasnya ruangan penyimpanan pada komputer. File PDF berisi informasi dalam bentuk file PDF dan banyak digunakan dalam dunia pendidikan, perkantoran, perusahaan dan lain sebagainya untuk membuat laporan, tugas, makalah, skripsi dan lainnya yang kemudian akan file PDF. Karena kebutuhan akan file PDF yang sangat penting, maka manusia cenderung mengumpulkan data berupa file PDF. Dalam proses bertukar data berupa file PDF baik secara online maupun offline sering kali tanpa disadari manusia mengirimkan dan kemudian menyimpannya dengan ukuran yang besar dan jelas hal ini menyebabkan kebutuhan akan media penyimpanan yang tidak sedikit. Adapun solusi untuk memecahkan masalah tersebut yaitu dengan melakukan kompresi. Hal ini bertujuan agar ukuran file PDF menjadi jauh lebih kecil sehingga proses transfer file PDF cepat serta dapat menghemat ruangan penyimpanan. Ada beberapa algoritma dalam mengkompresi file PDF, namun penelitian ini menggunakan algoritma Stout Code. Setalah melakukan kompresi pada algoritma tersebut, selanjutnya hasil kompresi dari algoritma tersebut yang berpatokan dengan beberapa parameter seperti Ratio Of Compression (RC), Compression Ratio (CR), Space Saving (SS). Hasil dari kompresi algoritma Stout Code maka akan dilakukan proses kompresi dan diperoleh file PDF hasil kompresi.

Kata Kunci: Kompresi; File PDF; Algoritma Stout Code

Abstract-*Human habits in collecting and exchanging data, one of which is in the form of PDF files, has an impact on limited storage space on computers. PDF files contain information in the form of PDF files and are widely used in the world of education, offices, companies and so on to create reports, assignments, papers, theses and others which will then be PDF files. Because the need for PDF files is very important, humans tend to collect data in the form of PDF files. In the process of exchanging data in the form of PDF files both online and offline, humans often unknowingly send and then store them in large and clear sizes, this causes the need for a large amount of storage media. The solution to solve this problem is to do compression. It is intended that the PDF file size becomes much smaller so that the PDF file transfer process is fast and can save storage space. There are several algorithms for compressing PDF files, but this study uses the Stout Code algorithm. After compressing the algorithm, then the results of the compression of the algorithm are based on several parameters such as Ratio Of Compression (RC), Compression Ratio (CR), Space Saving (SS). The results of the Stout Code compression algorithm will be compressed and a compressed PDF file will be obtained.*

Keywords: *Compression; PDF Files; Stout Code Algorithm.*

1. PENDAHULUAN

Kebiasaan manusia dalam mengumpulkan dan bertukar data salah satunya berupa *file* pdf memberikan dampak pada ruangan penyimpanan. *File* tersendiri berisi informasi dalam bentuk *file* pdf dan banyak digunakan dalam dunia pendidikan, perkantoran, perusahaan dan lain sebagainya. Untuk membuat laporan, tugas, makalah, skripsi dan lainnya yang kemudian disimpan dalam bentuk *file* pdf. Hal ini berupa *file* pdf memengaruhi besar *file* pdf . Ini terlihat bahwa kemungkinan ukuran *file* pdf maka proses bertukar dan memerlukan waktu yang lama, sebaliknya jika ukuran *file* pdf kecil maka proses bertukar data jauh lebih cepat.

Selain itu, ukuran *file* pdf yang besar juga akan menjadi masalah pada ruangan penyimpanan. Karena kebutuhan akan *file* pdf yang sangat penting, maka manusia cenderung mengunpulkan *file* data berupa *file* pdf dan seringkali tanpa disadari manusia menyimpan dan mengirimkannya dalam ukuran besar dan jelas, hal ini menyebabkan kebutuhan akan media penyimpanan yang tidak sedikit. Apalagi saat ini manusia lebih sering melakukan transfer *file* pdf baik secara *online* maupun *offline* yang tentunya akan membutuhkan waktu pengiriman yang cepat dan juga ruangan penyimpanan yang memadai. Adapun solusi untuk mengatasi masalah tersebut adalah dengan melakukan kompresi. Hal ini bertujuan agar ukuran *file* pdf menjadi jauh lebih kecil sehingga proses transfer *file* pdf lebih cepat serta dapat menghemat ruangan penyimpanan.[1]

Oleh karena itu, untuk mengatasi masalah tersebut digunakan kompresi data. Kompresi data merupakan salah satu kajian di dalam ilmu personal komputer yang bertujuan untuk mengurangi ukuran *file* sebelum meyimpan data tersebut ke pada media peyimpanan. Utama kompresi data terdiri dari dua proses yaitu kompresi serta dekompresi atau pemulihan data kembali seperti aslinya. Jika suatu *file* dikompresi, maka *file* tersebut wajib dapat kembali sesudah *file* tadi dikompresi[2].

Ada dua teknik yang dilakukan pada melakukan kompresi data yaitu *Lossless compression* dan *Compression Lossless. Lossles compression* artinya kompresi data dimana hasil dekompresi dari data yang terkompresi sama dengan data aslinya tidak hilang. Sedangkan *Lossy compression* artinya kompresi data dimana hasil kompresi dari data yang terkompresi tidak sama menggunakan data aslinya karena ada informasi yang hilang, tetapi masih dapat ditoleransi[3].

Jurnal Sains Dan Teknologi Informasi
Vol 4, No 1, Desember 2024, Hal 39-45
ISSN 2809-610X (Media Online)
DOI 10.47065/jussi.v4i1.7706
Website https://ejurnal.seminar-id.com/index.php/jussi

Jika terjadi kerangkapan *file* pdf atau karena telah terlalu banyak *file* dokumen, ruang penyimpanan menjadi lebih sedikit. Hal ini terjadi karena pada suatu media penyimpanan tidak hanya *file* dokumen saja yang ada, namun banyak *file-file* lainnya yang berbagi ruang penyimpanan. Bersumber pada penelitian sebelumnya dilaksanakan oleh Teguh Saputra ditahun 2021 tentang "Perancangan Aplikasi File Transfer Dengan Menerapkan Algoritma *Arithmetic coding*", dapat ditentukan algoritma *Arithmetic coding* akan berhenti melakukan proses kompresi apabila panjang *bit* karakter mencapai jumlah karakter terakhir dan algoritma tersebut dapat menangani proses untuk kompresi *file* transfer dengan cukup baik[4].

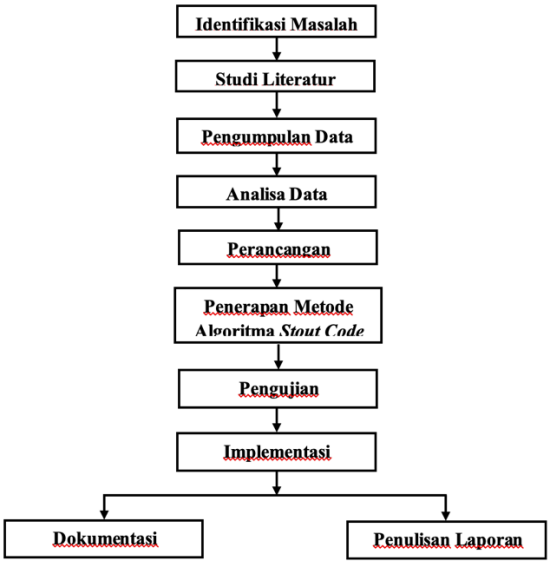
Berdasarkan penelitian yang dilakukan oleh M. Mawar pada tahun 2020 tentang "Perancangan Aplikasi Kompresi File Pdf Dengan Menerapkan Algoritma Punctured Elias Codes", ditentukan perancangan kompresi *file* pdf dengan menerapkan algoritma *Elias Codes* mempunyai skala kompresi 40,86% sehingga dapat memperkecil ukuran data suatu *file* pdf serta dapat menyesuikan jalur pemilihan[5].

Berdasarkan kinerja kompresi *file* pdf dengan menggunakan beberapa algoritma dari penelitian terdahulu, masing-masing algoritma menghasilkan kemampuan untuk meningkatkan proses pengiriman data dan bekaian terhadap ruang penyimpanan ini dikarenakan *bit* data yang telah berkurang sehingga mampu meningkatnya kecepatan dalam pemindahan *file* pdf. Sehingga dalam penelitian terdahulu sangat membantu dalam mempelajari kompresi pada *file* pdf.[6].

2. METODOLOGI PENELITIAN

2.1 Metodologi Penelitian

Untuk membantu dalam penyusunan penelitian ini, maka diperlukan peyususunan kerangka kerja yang dapat digunakan sebagai pendekatan dalam memecahkan masalah. Kerangka kerja ini merupakan langkah-langkah yang akan dilakukan dalam penyelesaian masalah yang akan dibahas. Adapun kerangka kerja yang penulis lakukan dapat dilihat pada gambar berikut:



Gambar 1. Kerangka Kerja Penelitian

Penjelasan dari kerangka kerja berdasarkan gambar 1 yang dilakukan dalam penelitian adalah sebagai berikut:

a. Mengidentifikasi Masalah, Pada tahap penelitian ini, yang dilakukan adalah mengidentifikasi masalah yang terjadi mengenai kompresi *file* pdf. Masalah yang ada terkait kaspasitas untuk keamanan data.
b. Studi Literatur, dilakukan untuk mencari data dan informasi yang berkaitan dengan kompresi *file* pdf dan metode stout code pada buku, jurnal beserta memberikan petunjuk dalam menyelesaikan permasalah yang ada.
c. Pengumpulan Data, dilakuan untuk memperoleh informasi yang dibutuhkan merangkai, mencapai tujuan penelitian untuk mengkompresikan *file* pdf.
d. Analisa data, Suatu proses untuk mengarang data yang telah membentuk bahan baru, hingga dapat dipakai untuk mengetahui apabila tercantum untuk mengkompresikan *file* pdf.
e. Perancangan, perancangan aplikasi menggunakan tools programman *Microsoft Visual Studio* 2010
f. Penerapan Metode *Stout Code,* Rancangan mengkompresi *file* pdf untuk mengecilkan ukuran kapasitas *file* pdf yang akan dikompresikan dengan menerapkan metode *Stout Code*.
g. Pengujian, selanjutnya adalah melakukan pengujian aplikasi yang sudah selesai. Pengujian dilakukan untuk melihat kesesuaian hasil yang diperoleh aplikas perancangan yang telah telah dibuat.
h. Implementasi, dilakukan untuk proses pembangunan perangkat lunak yang bertujuan untuk mengetahui apakah sistem dapat berjalan dengan baik dan sesuai dengan kebutuhan atau masih diperlukannya perbaikan pada sistem tersebut.

Jurnal Sains Dan Teknologi Informasi
Vol 4, No 1, Desember 2024, Hal 39-45
ISSN 2809-610X (Media Online)
DOI 10.47065/jussi.v4i1.7706
Website https://ejurnal.seminar-id.com/index.php/jussi

i. Dokumentasi, dilakukan untuk tujuan dikemukakan dari buatan penerapan metode stout code untuk mengkompresikan *file* pdf beserta penelitian yang akan dijadikan.
j. Penulisan Laporan, mendokumentasikan seluruh kegiatan hasil penelitian dalam bentuk skripsi yang nantinya juga dibuat dalam bentuk artikel ilmiah yang akan dipublikasikan.

2.2 Algoritma Stout code

Dalam *Stout Code*, panjang variabel untuk kode bilangan bulat mirip dengan kode *Elias Omega* dan *Even-Rodeh Code*. Codeword yang dihasilkan oleh algoritma *Stout Code* bergantung pada pilihan parameter l yang lebih besar atau sama dengan 2[11]. Algoritma *Stout Code* diperkenalkan oleh *Quentin Stout* dengan dua keluarga yaitu Rl dan Sl rekursif. Dalam keluarga Rl, semakin banyak kelompok yang dibaca sampai kelompok tersebut ditemukan dan diikuti dengan 0. Gunakan notasi L = 1 + ? Log2 n? dan dilambangkan dengan representasi biner B (n, l) dari l – bit (kode beta) dari bilangan bulat n. Contohnya, B(12, 5) = 01100. Untuk l ≥ 2.

Kedua dari *Stout Code* memiliki metode yang serupa, tetapi dengan awalan berbeda yang dilambangkan dengan Sl(n). Untuk nilai kecil 1, kelompok ini menawarkan beberapa peningkatan dibandingakan dengan kode Rl. Khususnya ini menghilangkan sedikit redundansi dalam kode Rl karena panjang kelompok tidak boleh 0. Awalan Sl mirip dengan awalan Rl dengan perbedaanya bahwa panjang kelompok untuk L mengkodekan L – 1 – l. Awalan Sl(n). Dalam membangun codeword dari algoritma Stout Code pertama sekali menentukan nilai l, pada penelitian ini menggunakan nilai l = 2. Kemudian setelah menentukan nilai l. Langkah-langkah encoding algoritma *Stout Code* adalah sebagai berikut:

a. Menyiapkan *file* pdf yang akan dikompresi.
b. Buka software HxD, kemudian input *file* pdf yang akan dikompresi untuk mendapatkan nilai *hexadecimal.*
c. Mengambil 16 nilai *hexadecimal* sebagai sampel.
d. Buatlah tabel tampilan nilai bilangan *hexadesimal* sample
e. Urutkan nilai bilangan *hexadecimal* berdasarkan frekuensi kemunculan terbanyak, frekuensi kemunculan paling banyak akan berada diurutan paling atas.
f. Buatlah tabel nilai *hexadecimal* yang belum dikompresi, dari tabel ini kita bisa mengetahui ukuran jumlah bit awal sebelum dikompresi.
g. Selanjutnya, proses kompresi nilai *hexadecimal file* pdf dengan algoritma Stout Code dilakukan dengan cara menggantikan nilai biner menggunakan nilai kode S2(n) algoritma *Stout Code*.
h. Buatlah tabel nilai *hexadecimal* yang sudah di kompresi dengan kode S2(n) algoritma *Stout Code* sehingga dapat diketahui berapa hasil kompresinya dari jumlah bit awal.
i. Menyusun *string bit* dapat membuat operasi kompresi sebanding melalui tempat nilai ketentuan *hexadecimal* awal sebelum di kompresi.
j. Buatlah tabel string bit dari hasil kompresi, dengan cara menyusun codeword S2(n) sesuai dengan susunan nilai hexadecimal awal.
k. Menambahkan *padding* dan *flagging*.
l. Setelah ditambahkan *padding* dan *flaging* pada *string bit*, tahapan berikutnya adalah membagi perdelapan bit dari seluruh *string bit* kemudian menggelompokkannya.
m. Buatlah tabel pengelompokkan *string bit* setalah ditambahkan *padding* dan *flagging.*
n. Berdasarkan tabel pengelompokkan *string bit* ubah hitungan biner ke hitungan *hexadecimal* buat mengenal karakter dari hasil melalui tanda ASCII.
o. Buatlah tabel hasil karakter terkompresi.

Langkah-langkah decoding algoritma *Stout Code* adalah sebagai berikut:

a. Menyiapkan *string bit* yang sudah didapatkan dari proses kompresi diatas.
b. Hilangkan *padding* dan *flaging*, dengan cara menghilangkan *bit* pada bagian akhir sebanyak 7 + n.
c. Setelah dilakukan langkah kedua tersebut maka akan mendapatkan *string bit* hasil kompresi semula, kemudian selanjutnya lakukan pembacaan *string bit* awal.
d. Setelah melakukan proses diatas kemudian buat tabel pencocokan *bit*[12].

3. ANALISA DAN PEMBAHASAN

3.1 Proses Kompresi *File* Pdf Menerapakan Algoritma Stout Code

Langkah-langkah yang dilakukan dalam melakukan kompresi *file* pdf menggunakan agoritma *Stout Code* adalah sebagai berikut:

a. Mengambil 16 nilai *hexadesimal* sample *file* pdf
Berdasarkan sample nilai *hexadesimal file* pdf di atas, maka diambil 16 sample nilai *hexadesimal* untuk proses kompresi *file* pdf.

Tabel 1. Tampilan Nilai Bilangan *Hexadesimal*

25	50	44	46	2D	31	2E	37
0D	0A	25	A1	B3	C5	D7	0D

Jurnal Sains Dan Teknologi Informasi
Vol 4, No 1, Desember 2024, Hal 39-45
ISSN 2809-610X (Media Online)
DOI 10.47065/jussi.v4i1.7706
Website https://ejurnal.seminar-id.com/index.php/jussi

b. Mengurutkan nilai *hexadesimal file* pdf berdasarkan frekuensi kemunculan
hingga terdapat jumlah hitungan *hexadesimal* yang dapat untuk memperkirakan secara manual. Jumlah hitungan *hexadesimal* disusun dari waktu kemunculan dimulai dari frekuensi terbesar ke terkecil nilai *hexadesimal* yang memiliki banyak frekuensi yang sudah disusun diatas.

Tabel 2. Nilai *Hexadesimal* yang belum dikompresi

No	Nilai Hexadesimal	Nilai Biner	Bit	Frekuensi	Bit x Frekuensi
1	25	00100101	8	2	16
2	0D	00001101	8	2	16
3	50	01010000	8	1	8
4	44	01000100	8	1	8
5	46	01000110	8	1	8
6	2D	00101101	8	1	8
7	31	00110001	8	1	8
8	2E	00101110	8	1	8
9	37	00110111	8	1	8
10	0A	00001010	8	1	8
11	A1	10100001	8	1	8
12	B3	10110011	8	1	8
13	C5	11000101	8	1	8
14	D7	11010111	8	1	8
		Total Bit			**128 Bit**

Berdasarkan tabel di atas, satu nilai *hexadesimal* terdapat delapan bit bilangan biner. Maka 16 nilai *hexadesimal* mempunyai nilai biner sebanyak 128bit.

c. Pembentukan *Codeword* S2(n) pada algoritma *Stout Code*
Aturan dalam pembentukan *Codeword* S2(n) dengan menggunakan algoritma *Stout Code* dapat dilihat pada landasan teori pada sub bab sebelumnya. Adapun *Codeword* S2(n) pada algoritma *Stout Code* dapat dilihat pada tabel berikut :

Tabel 3. Codeword S2(n)

L	S2(n)	S3(n)
1	01	001
2	10	010
3	11	011
4	00 100	100
5	00 101	101
6	00 110	110
7	00 111	111
8	01 1000	000 1000
9	01 1001	000 1001
10	01 1010	000 1010
11	01 1011	000 1011
12	01 1100	000 1100
13	01 1101	000 1101
14	01 1110	000 1110
15	01 1111	000 1111
16	10 10000	001 10000

Berdasarkan tabel codeword S2(n) pada algoritma *Stout Code* diatas hanya menunjukan kode sampai karakter ke 16. Bagaimana proses yang dilakukan untuk mendapatkan kode S2(n), jika jumlah karakter yang dihasilkan lebih dari 16 karakter. Contoh n = 17 proses awalnya yang dilakukan adalah dengan menentukan nilai dari I = 2, maka nilai biner dari 17 adalah 10001, dengan panjang bit yaitu L = 5 selanjutnya mencari nilai R2 (5-1-2) = 2adalah 10 dan diambil sebanyak nilai I, yaitu 2 nilai R = 10 sehingga R (L-1-I) B(n,L) = 1010001.

d. Kompresi nilai *hexadesimal* file pdf dengan codeword S2(n) pada algoritma *Stout Code*. Proses selanjutnya adalah melakukan kompresi nilai *hexadesimal* S2(n) pada algoritma *Stout Code* pada tabel berikut in

Tabel 4. Nilai *Hxadesimal* yang sudah dikompresi dengan codeword S2(n) pada algoritma *Stout Code*

No.	Nilai Hexadesimal	Codeword S2(n)	Bit	Frekuensi	Bit x Frekuensi
1	25	01	2	2	4
2	0D	10	2	2	4

Jurnal Sains Dan Teknologi Informasi
Vol 4, No 1, Desember 2024, Hal 39-45
ISSN 2809-610X (Media Online)
DOI 10.47065/jussi.v4i1.7706
Website https://ejurnal.seminar-id.com/index.php/jussi

No.	Nilai Hexadesimal	Codeword S2(n)	Bit	Frekuensi	Bit x Frekuensi
3	50	11	2	1	2
4	44	00100	5	1	5
5	46	00101	5	1	5
6	2D	00110	5	1	5
7	31	00111	5	1	5
8	2E	011000	6	1	6
9	37	011001	6	1	6
10	0A	011010	6	1	6
11	A1	011011	6	1	6
12	B3	011100	6	1	6
13	C5	011101	6	1	6
14	D7	011110	6	1	6
		Total Bit x Frekuensi			**72 Bit**

e. Penyusunan *String Bit*
Setelah proses kompresi nilai *hexdesimal file* pdf dengan algoritma *Stout Code* menerapkan nilai kode S2(n), lalu seterusnya mengurutkan balik *string bit* yang telah dihasilkan mulai operasi kompresi dapat dilihat dari tabel berikut.

Tabel 5. *String Bit* Hasil Kompresi Menerapkan Algoritma *Stout Code*

25	**50**	**44**	**46**
01	11	00100	00101
2D	**31**	**2E**	**37**
00110	00111	011000	011001
0D	**0A**	**25**	**A1**
10	011010	01	011011
B3	**C5**	**D7**	**0D**
011100	011101	011110	10
Total 72 *Bit*			

Berdasarkan tabel 5 *string bit* yang dihasilkan dari proses kompresi file pdf menerapkan algoritma *Stout Code* dapat dilihat pada tabel berikut:

Tabel 6. *String Bit* Hasil Kompresi

01	11	00100	00101
00110	00111	011000	011001
10	011010	01	011011
011100	011101	011110	10

f. Pemeriksaan *String Bit*
String Bit yang dihasilkan dari hasil kompresi yang sudah digabungkan adalah sebagai berikut:
"01110010000101001100011101100001100110011010010110110111000111010111101 0" dengan total 72 *bit*. Selanjutnya, *string bit* dibagi per 8 *bit* atau 72 mod 8 = 0 dan dinyatakan dengan simbol n. Kemudian dilakukan pemeriksaan *string bit,* untuk pemeriksaan *string bit* ada beberapa langkah yaitu :
1. Jika sisa bagi = 0, maka tambahkan 00000001, dinyatakan sebagai *bit* akhir
2. Jika sisa bagi n (1,2,3,4,5,6,7), maka tambahkan *padding* dan *flaging*. Rumus *paddin* adalah 7 – n +"1" artinya tambahkan "0" sebanyak 7 – n +"1" di akhir *ztring bit* dan dinyatakan dengan L. Lalu tambahkan bilangan biner dari hasil 9 – n dan dinyatakan sebagai *bit* akhir.

g. Penambahan *Padding* dan *Flagging*
Hasil dari pemeriksaan *string bit* yang bernilai 72 habis dibagi 8 atau sisa bagi adalah 0 karena bilangan 8 bahwa bukan harus dilakukan penambahan *padding* dan *flagging*, tapi perlu meneruskan 00000001 pada *bit* akhir. Jadi *string bit* yang terbentuk setelah penambahan 00000001 pada *bit* akhir adalah sebagai berikut:
"01110010,00010100,11000111,01100001,10011001,10100101,10110111,00011101,01111010,**00000001** sehingga total panjang *bit* adalah 80.

h. Pengelompokan *String Bit*
Setelah pengembangan *padding* dan *flagging* lalu terbentuklah *string bit* yang baru dan proses selanjutnya dilakukan pengelompokkan *string bit* membentuk selama kelompok dimana setiap kelompok berlaku dari 8 bit

Tabel 7. Pengelompokan *Bit*

01110010	00010100	11000111	01100001
10011001	10100101	10110111	00011101
01111010	**00000001**		

Jurnal Sains Dan Teknologi Informasi
Vol 4, No 1, Desember 2024, Hal 39-45
ISSN 2809-610X (Media Online)
DOI 10.47065/jussi.v4i1.7706
Website https://ejurnal.seminar-id.com/index.php/jussi

i. Hasil Kompresi Algoritma Stout Code
Berdasarkan pada hasil pengelompokkan *bit* di atas, maka terbentuklah 9 kelompok hasil jumlah biner yang baru siap dikompresi bersama penambahan jumlah biner. Sesudah dikelompokan lalu sseterusnya mengantikan nilai biner ke nilai *hexadesimal*, buat mengeenal karakter telah dihasilkan dapat dilihat dari tabel berikut.

Tabel 8. Hasil Karakter Terkompresi

No	Biner	*Decimal*	*Hexadesimal*	Karakter	Keterangan
1	01110010	114	72	R	Huruf latin r kecil
2	00010100	20	14		Device control 4 (tidak terlihat)
3	11000111	199	C7	Ç	Latin capital letter C with cedilla
4	01100001	97	61	A	Huruf latian a kecil
5	10011001	153	99	™	Trade mark sign
6	10100101	165	A5	¥	Tanda Yen
7	10110111	183	B7	·	Middle dot
8	00011101	29	1D		Grop seperator
9	01111010	122	7A	Z	Huruf latin z kecil
10	**00000001**	1	1		Star of Heading (tidak terlihat)

Setelah dilakukan proses kompresi pada file pdf menggunakan algoritma Stout Code maka hasil dari proses kompresi menghasilkan karakter R Ç A™¥. Z karakter hasil kompresi tersebut dikembalikan dalam bentuk file pdf. Setelah itu buat mengketahui kinerja algoritma *Sotout* Code kompresi, dan untuk memperkirakan kinerja perhitungan kompresi *file* pdf tercantum pada parameter bahwa sudah ditentukan. makin minim perhitungan kompresi dan berkelas kinerja kompresi, beserta parameter untuk memperkirakan hasil kinerja algoritma *Stout Code*.

a. *Ratio of Compression* (RC)

$$RC = \frac{Ukuran\ Data\ Sebelum\ Dikompresi}{Ukuran\ Data\ setelah\ Dikompresi}$$

$$RC = \frac{128}{72} = 1{,}77$$

b. *Compression Ratio* (CR)

$$CR = \frac{Ukuran\ Data\ Sebelum\ Dikompresi}{Ukuran\ Data\ setelah\ Dikompresi} \times 100\%$$

$$CR = \frac{72}{128} \times 100\%$$

$$CR = 56{,}25\%$$

c. *Space Saving*

$Ss = 100\% - CR$

$Ss = 100\% - 50\%$

$Ss = 50\%$

Mulai hasil hitungan diatas diperoleh ketentuan maka hasil kompresi *File* pdf dengan menerapkan algoritma *Stout Code* adalah sebagai berikut:

CR = Ukuran file pdf awal x Compression Ratio
= 3,40MB x 50% = 1,7MB

CR = Ukuran file pdf awal – Hasil perkalian diatas
= 3,40MB – 1,7 MB
= 1,7MB

4. KESIMPULAN

Berdasarkan hasil dari analisa yang telah dilakukan, maka penulis mengambil kesimpulan dari penelitian ini adalah Proses kompresi *file* pdf dengan algoritma *Stout Code* dilakukan pencarian *file* pdf kemudian dilakukan proses kompresi dan dekompresi sehingga diperoleh hasil yang lebih kecil. Perancangan kompresi *file* pdf dengan algoritma *Stout Code* dibangun dengan microsoft visual studio 2010. Penerapan algoritma *stout code* untuk kompresi *file* pdf dilakukan dengan aplikasi HxD untuk membaca nilai *hexadesimal* dari suatu *file* pdf, lalu mengubah nilai *heksa* tersebut menjadi nilai bit yang baru berupa bilang biner, selanjutnya menyusun kembali nilai biner tersebut dan menjadi karakter yang baru. Sistem yang dirancang dalam penelitian ini masih memiliki kekurangan sehingga dapat dikembangkan dan diperbaiki lebih lanjut. Dalam penerapan algoritma dapat dikembangkan lagi dengan penambahan objek seperti *file* dokumen, *file* gambar, *file* audio, dan lain-lain sehingga memberikan manfaat yang lebih baik dimasa yang akan mendatang. Dalam penelitian ini, perancangan aplikasi kompresi *file* pdf hanya dapat mengkompresi pdf yang berektensi dan diharapkan kedepannya bisa dapat mengkompresi seperti *file* dokumen seprti gambar, Mp3 serta komponen lainnya.

Jurnal Sains Dan Teknologi Informasi
Vol 4, No 1, Desember 2024, Hal 39-45
ISSN 2809-610X (Media Online)
DOI 10.47065/jussi.v4i1.7706
Website https://ejurnal.seminar-id.com/index.php/jussi

REFERENCES

[1] A. Apijuddin Kompresi and A. S. Codes, "Perancangan Aplikasi Kompresi File Teks Menggunakan Algoritma Stout Codes," vol. 9, pp. 183–188, 2021.

[2] Lamsah and D. P. Utomo, "Penerapan Algoritma Stout Codes Untuk Kompresi Record Pada Databade Di Aplikasi Kumpulan Novel," *KOMIK (Konferensi Nas. Teknol. Inf. dan Komputer)*, vol. 4, no. 1, pp. 311–314, 2020, doi: 10.30865/komik.v4i1.2710.

[3] A. S. Sinaga and E. Buulolo, "Perancangan Aplikasi Kompresi File Animasi Flash FLA Dengan Menggunakan Algoritma Stout Codes," *KOMIK (Konferensi Nas. …*, vol. 4, pp. 116–120, 2020, doi: 10.30865/komik.v4i1.2650.

[4] T. Saputra, "Perancangan Aplikasi File Transfer Dengan Menerapkan Algoritma Arithmetic Coding," vol. 9, no. April, pp. 289–295, 2021.

[5] Mawar, "Perancangan Aplikasi Kompresi File Pdf Dengan Menerapkan Algoritma Punctured Elias Codes Mawar," *Inf. dan Teknol. Ilm.*, vol. 7, no. 3, pp. 217–223, 2020, [Online]. Available: http://ejurnal.stmik-budidarma.ac.id/index.php/inti/article/view/2391

[6] H. S. Purba, "Implementasi Algoritma Stout Code untuk Kompresi Record Database pada Website Dinas Pariwista dan Kebudayaan Kabupaten Dairi," vol. 10, no. April, pp. 129–133, 2022.

[7] R. Yanur Tanjung, "Perancangan Aplikasi Kompresi File Dokumen Menggunakan Algoritma Adiitive Code," *J. Ris. Komputer)*, vol. 8, no. 4, pp. 2407–389, 2021, doi: 10.30865/jurikom.v8i4.3593.

[8] T. Sasono and I. Indriyani, "Upaya Proses Pemampatan Produksi Biogas Untuk Peningkatan Kapasitas Storage," *J. Tek. Energi*, vol. 11, no. 2, pp. 30–35, 2022, doi: 10.35313/energi.v11i2.3717.

[9] J. A. Simatupang, G. L. Ginting, and F. T. Waruwu, "Perancangan Aplikasi Kompresi File Video Dengan Menggunakan Algoritma Start Step Stop Code," vol. 1, no. 2, pp. 41–48, 2022.

[10] D. Cahayati, A. M. H. Pardede, and H. Khair, "Implementasi Algoritma Elias Gamma Kompresi Pada File Teks," vol. 6341, no. April, pp. 159–166, 2022.

[11] M. Mesran and S. D. Nasution, "Peningkatan Keamanan Kriptografi Caesar Cipher dengan Menerapkan Algoritma Kompresi Stout Codes," *J. RESTI (Rekayasa Sist. dan Teknol. Informasi)*, vol. 4, no. 6, pp. 7–12, 2020, doi: 10.29207/resti.v4i6.2730.

[12] T. D. Lumbantobing, N. A. Hasibuan, and S. A. Hutabarat, "Implementasi Algoritma Gabor Wavelet Dalam Pengenalan Sketsa Wajah Pada Citra Digital," vol. 8, no. 5, pp. 170–176, 2021, doi: 10.30865/jurikom.v8i5.3632.

[13] S. D. Manullang, E. Buulolo, and I. Lubis, "Implementasi Data Mining Dalam Memprediksi Jumlah Pinjaman Dengan Algoritma C4.5 Pada Kopdit CU Damai Sejahtera," *J. Sist. Komput. dan Inform.*, vol. 1, no. 3, p. 265, 2020, doi: 10.30865/json.v1i3.2153.

[14] D. Iqbal, "Implementasi Algoritma Levenstein Untuk Kompresi File Video Pada Aplikasi Chatting Berbasis Android," *KOMIK (Konferensi Nas. Teknol. Inf. dan Komputer)*, vol. 3, no. 1, pp. 266–273, 2019, doi: 10.30865/komik.v3i1.1601.

[15] T. P. Sari, S. D. Nasution, and R. K. Hondro, "Penerapan Algoritma Levenstein Pada Aplikasi Kompresi File Mp3," *KOMIK (Konferensi Nas. Teknol. Inf. dan Komputer)*, vol. 2, no. 1, 2018, doi: 10.30865/komik.v2i1.946.