Informasi dan Teknologi Ilmiah (INTI)
ISSN 2301-9425 (Media Cetak), ISSN 2798-0340 (Media Online)
Volume 10, No 3, Juni 2023
Hal: 104-111

Perbandingan Algoritma Punctured Elias Code Dan Stout Code Untuk Kompresi File PDF

Vindi Aulia Sari*, Lince Tomoria Sianturi, Chandra Frenki Sianturi

Fakultas Ilmu Komputer dan Teknologi Informasi, Teknik Informatika, Universitas Budi Darma, Medan, Indonesia
Email: [1]vindiauliasari12@gmail.com, [2]lince.sianturi338@gmail.com
Email Penulis Korespondensi: vindiauliasari12@gmail.com

Abstrak−Saat ini, penggunaan aplikasi sebagai informasi semakin umum digunakan. Tetapi ada masalah yang sering dijumpai, yaitu kebutuhan akan tempat besar sebagai media penyimpanannya. Oleh sebab itu, dibutuhkan kompresi untuk memperkecil ukuran data aplikasi tersebut. Ada beberapa algoritma dalam melakukan proses kompresi. Dari banyaknya algoritma tersebut maka pengguna tidak mengetahui algoritma mana yang leih akurat dalam proses kompresi. Dari banyaknya algoritma kompresi maka perlu dilakukan proses perbandingan algoritma-algoritma kompresi tersebut agar diketahui algoritma mana yang lebih akurat dalam melakukan proses kompresi file pdf. Dilakukan perbandingan algoritma maka akan didapatkan algoritma yang lebih akurat seperti perbandingan algoritma punctured elias code dan stout code dalam mengkompresi file pdf didapatkan bahwa rasio kompresi algoritma punctured elias code lebih besar daripada algoritma stout code dengan rasio kompresi 75%. Besarnya nilai rasio kompresi maka semakin akurat atau baik algoritma tersebut dalam melakukan proses kompresi.

Kata Kunci: Kompresi; Perbandingan; Punctured Elias Code; Stout Code

Abstrak−Currently, the use of applications as information is increasingly commonly used. But there is a problem that is often encountered, namely the need for a large place as a storage medium. Therefore, compression is needed to reduce the size of the application data. There are several algorithms in performing the compression process. Of the many algorithms, the user does not know which algorithm is more accurate in the compression process. Of the many compression algorithms, it is necessary to compare the compression algorithms in order to find out which algorithm is more accurate in compressing PDF files. By comparing the algorithms, a more accurate algorithm will be obtained, such as a comparison of the punctured elias code and stout code algorithms in compressing pdf files, it is found that the compression ratio of the punctured elias code algorithm is greater than the stout code algorithm with a compression ratio of 75%. The greater the value of the compression ratio, the more accurate or better the algorithm is in carrying out the compression process.

Keywords: Compression; Comparison; Puctured Elias Code; Stout Code

1. PENDAHULUAN

Kompresi data ialah suatu cara dalam ilmu komputer untuk pengecilan ukuran data ataupun memampatkan data. Data yang akan dimampatkan berukuran lebih kecil dari data sebelumnya dengan tujuan agar terjadi penghematan penyimpanan. Jika kompresi data dilakukan, maka ruang penyimpanan yang dibutuhkan kecil. Selain lebih efisien, kompresi juga dapat menyebabkan lebih cepatnya waktu pertukuran data. Seiring berkembangnya teknologi pada zaman sekarang ini, data memiliki peranan yang sangat penting, besar jumlah data yang disimpan padad memori atau hardisk akan terjadi penambahan besar kapasitas terpakai media penyimpanan.

Ukuran file pdf tergolong sangat besar dan terjadi pengaruh pada ruang penyimpanan serta waktu berkirim data juga tergolong lama dengan ukuran file yang begitu besar. Oleh karenanya perlu dilakukan kompresi dalam memampatkan isi file pdf. Ketika proses kompresi dilakukan harus ada algoritma pendukung dalam kompresi. File pdf lebih fleksibel dibandingkan file dokumen lainnya seperti docx dan rtf sebab file pdf dapat diakses diperangkat mana saja. Serta file pdf memberikan informasi yang asli dan sesuai tanpa harus khawatir file berantakan ketika diakses diperangkat mana saja.

Pada penelitian kali dilakukan perbandingan algoritma untuk mengetahui algoritma manakah yang akurat dalam proses komperesi file pdf. Algoritma yang akan dibandingkan yaitu algoritma Punctured Elias Code dan Stout Code. Kedua algoritma tersebut adalah algoritma untuk proses kompresi dan variabel perbadingan kedua algoritma adalah rasio kompresi dan space saving.

Punctured adalah sebuah bilangan integer pada suatu percobaan dalam peningkatan performa the Burrows-Wheeler transform. Istilah punctured berasal dari pengawasan eror kode-kode (ECC). ECC terdiri atas data asli ditambah sebagian bilangan check bits. Apabila beberapa check bits dihapus, untuk memperingkas rangkaian kode, hasil kode dituju sebagai punctured. Kode punctured ini diberi nama kode P1 serta kode tersebut dimulai dari 1 (paling sedikit terdapat satu flag, kecuali untuk kode P1) dan juga diakhiri dengan 1 (karena n yang asli, yaitu bit MSB (Most Significant Bit) adalah satu, telah dibalikkan) [1]. Sedangkan Stout Code adalah salah satu algoritma kompresi dimana dalam makalah pendeknya Quentin stout diperkenalkannya dua keluarga RI serta SI pada rekursi, kode panjang variabel dalam bilangan bulat, mirip dengan serta lebih umum pada Elias Omega serta Even – Rodeh Code. Stout Code memiliki sifat universal serta asimtotik optimal dari Bilangan bulat n diberikan oleh L =1 + log2 n.

Penelitian yang dilakukan oleh Apujudin dalam skripsinya yang berjudul "Perancangan Aplikasi Kompresi File Teks Menggunakan Algoritma Stout Code" mengatakan bahwa file teks yang dikompresi dengan algoritma Stout Code menghasilkan sebuah file terkompresi yang memiliki ukuran lebih kecil[3]. Penelitian lainnya juga dilakukan oleh Dedek Andri Yansyah dengan judul penelitian "Perbandingan Metode Punctured Elias Code da n

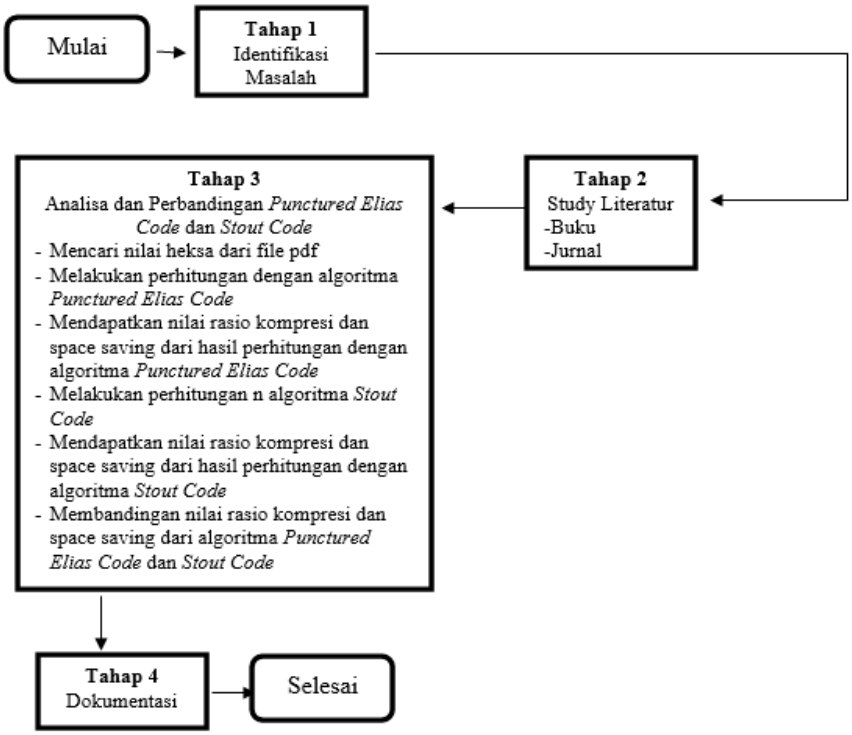
Huffman Pada Kompresi File Text" menyimpulkan bahwa pada kompresi dengan memakai metode Huffman lebih optimal apabila variasi karakter pada informasi tersebut tak terlalu banyak walau frekuensi munculnya tinggi sebab pohon Huffman yang akan terbentuk tak terlalu panjang hingga kode Huffman yang mewakilli karakter tersebut menjadi lebih singkat[4]. Penelitian lainnya yang dilakukan oleh Yuni Dewianingsih dengan judul penelitian " Kompresi File Aplikasi Hadist Dengan Menggunakan Algoritma Punctured Elias Code" dan kesimpulan yang didapatkan dari penelitian tersebut adalah hasil dari kompresi dengan menggunakan Punctured Elies Code mendapatkan rasio sebesar 68,75%[5]. Penelitian lainnya yang dilakukan oleh Lamsah dengan judul penelitian "Penerapan Algoritma Stout Code Untuk Kompresi Record Pada Database di Aplikasi Kumpulan Novel" mendapatkan kesimpulan bahwa dengan kompresi file record database hasil yang didapatkan ialah ukuran file lebih kecil dari ukuran awal serta dapat dipakai dalam alternatif penghemat media penyimpanan[6].

2. METODOLOGI PENELITIAN

2.1 Tahapan Penelitian

Pada metodologi penelitian dilakukan penjabaran tahapan yang dipakai pada penelitian. Metodologi penelitian terdiri atas tahapan yang memiliki kaitan dengan sistematis. Tahapan ini perlu dalam mempermudah saat dilakukan penelitian. Sebelum membuat kerangka penelitian, penulis terlebih dahulu menganalisa topik yang akan diteliti.

a. Tahap Identifikasi Masalah
   Pada tahap ini merupakan cara dari penulis untuk dapat menduka, memperkirakan dan menguraikan apa saja yang sedang menjadi masalah pada perbandingan algoritma *Punctured Elias Code* dan *Stout Code* dalam melakukan kompresi.
b. Tahap *Study* Literatur
   Pada penelitian pengumpulan data yang berkaitan dengan algoritma *Punctured Elias Code* dan *Stout Code*. Penulis mengumpulkan sebagai referensi baik dari buku, jurnal dan situs internet yang memiliki keterkatian dengan penelitian.
c. Analisa
   Tahapan analisa digunakan untuk mengetahui apa yang menjadi sumber masalah pada perbandingan algoritma *Punctured Elias Code* dan *Stout Code*, sehingga penyelesaian yang dihasilkan nantinya dapat mengatasi permasalahan yang ada.
d. Dokumentasi
   Tahap dilakukan dengan pendokumentasian hasil analisa dan pengujian secara tertulis dalam bentuk laporan penelitian.



Gambar 1. Flowchart Tahapan Penelitian

2.2 Kompresi

Kompresi adalah cara pengecilan ataupun pemampatan *file* yang memiliki ukuran besar menjadi lebih kecil serta

berkurangnya kebutuhan ruang penyimpanan. Proses kompresi ialah proses yang mengarah pada minimasi jumlah bit dalam representasi digital misalnya gambar, audio serta vidio, yang ukuran data jadi lebih kecil tetapi kualitas informasi tetap terjaga[3]. Kompresi *file* pdf ialah cara mengecilkan jumlah tiap bit yang direpresentasikan dalam *file* pdf dengan data yang jadi lebih kecil. Kompresi data memiliki arti teknik memampatkan data dan didapatkan ukuran data lebih kecil dari ukuran awal serta menjadi efisien dalam menyimpan ataupun waktu menukar data lebih singkat[3]

2.3 Algoritma Punctured Elias Code

Punctured Elias Codes pad bilangan bulat dibuat *Peter Fenwick* pada sebuah percobaan dalam ditingkatkannya *the Burrows–Wheeler transform*. Istilah *Punctured* berasal dari tempat pengawas eror kode-kode (ECC). ECC terdiri atas data asli ditambahkan jumlah bilangan pada *check bits*. Apabila *check bits* dihilangkan, agar dipersingkatnya serangkaian kode itu, hasil kode disebut *Punctured*[7]. Langkah dalam membangun kode *Punctured Elias Codes* ialah sebagai berikut:

1. Ambil bilangan biner dari n,
2. Reversed (balikkan bit-bitnya), serta siapkan *flag* dalam melihat jumlah bit yang memiliki nilai 1 pada n.
3. Pada setiap bit 1 pada n disiapkan *flag* dari 1 serta akhir *flag* bernilai 0.
4. Dilakukan penggabungan *flag* menggunakan bilangan biner yang telah *reserved*.

Tabel 1. *Punctured Elias Code*

N	Binary Of n	Reserved	*Flag*	*Flag* | Reserved	P1
0	0	-	-	-	0
1	1	1	10	10 | 1	101
2	10	01	10	10 | 01	1001
3	11	11	110	110 | 11	11011
4	100	001	10	10 | 001	10001
5	101	101	110	110 | 101	110101
6	110	011	110	110 | 011	110011
7	111	111	1110	1110 | 111	1110111
8	1000	0001	10	10 |0001	100001

2.4 Algoritma Stout Code

Keluarga kedua kode Stout serupa, tetapi dengan awalan yang berbeda dilambangkan dengan Sl (n). Untuk nilai l kecil, keluarga ini menawarkan beberapa peningkatan dibandingkan kode Rl. Secara khusus, ini menghilangkan sedikit redundansi yang ada dalam kode Rl karena grup panjang tidak boleh 0 (itulah sebabnya grup panjang dalam kode omega mengkodekan Li − 1 dan bukan Li). Awalan Sl mirip dengan awalan Rl dengan perbedaan bahwa grup panjang untuk Li mengkodekan Li −1 − l. Jadi, S2 (985), awalan dari bilangan bulat 985-bit n, dimulai dengan grup panjang 10-bit 1111011001 dan menambahkannya ke grup panjang untuk 10−1−2 = 7 = 1112. Untuk ini didahului grup panjang untuk 3−1−2 = 0 sebagai dua bit 00. Hasilnya adalah awalan 15-bit 00 | 111 | 1111011001, lebih pendek dari 19 bit R2 (985). Contoh lain adalah S3 (985), yang dimulai dengan 1111011001 yang sama dan bergantung padanya kelompok panjang untuk 10−1−3 = 6 = 1102. Rekursi berhenti pada titik ini karena 110 adalah grup l-bit. Hasilnya adalah 13-bit *codeword* 110 | 1111011001, sekali lagi lebih pendek dari 17 bit R3 (985). Awalan Sl (n) didefinisikan secara rekursif oleh:

$$S_l(n) = \begin{cases} B(n,l), & \text{for } 0 \le n \le 2^l - 1 \\ R_l(L_{-l}) B(n,L), & \text{for } n \ge 2^l \end{cases}$$

Tabel 1. berisi daftar beberapa awalan S2 (n) dan S3 (n) dan menggambarkan keteraturannya. Melihat bahwa kolom paling kiri mencantumkan nilai L, mis., panjang bilangan bulat yang dikodekan, dan bukan bilangan bulat itu sendiri. Grup panjang mempertahankan nilainya hingga grup yang mengikutinya menjadi semua 1, di mana titik grup bertambah 1 dan grup yang mengikuti diatur ulang ke 10 ... 0. Semua grup panjang, kecuali mungkin yang paling kiri, mulai dengan 1. Perilaku reguler ini adalah hasil dari pilihan Li −1 − l.

Tabel 2. Kode S2(n) dan Kode S3(n)

L	$S_2(n)$	$S_3(n)$
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

3. HASIL DAN PEMBAHASAN

Tahap analisa ialah langkah yang memiliki pengaruh serta penentuan untuk langkah berikutnya. Analisa pada sistem ialah langkah yang begitu penting dalam diketahuinya proses yang terjadi pada aplikasi yang dirancang saat melakukan perbandingan kompresi, dalam hal ini peneliti menggunakan algoritma *Punctured Elias Code dan Stout Code* untuk kompresi *file* PDF. Perbandingan algoritma digunakan agar diketahuinya algoritma mana lebih akurat pada pengecilan ukuran *file* PDF. Pengompresan *file* PDF berguna dalam dikuranginya ruang penyimpanan dan dipercepatnya proses dikirimnya *file*. Pada penelitian ini hal paling utama ialah membandingan besarnya rasio kompresi serta *space saving* dari algoritma *Punctured Elias Code dan Stout Code*.

File dokumen yang berekstensi PDF mempunyai ukuran yang cukup besar, semakin lama waktu pada *file* dokumen maka makin besar pula ruang penyimpanan yang diperlukan, dan proses pengiriman *file* tergolong cukup lama. Saat dilakukannya proses kompresi *file* PDF harus dilakukan terlebih dahulu analisa pada *file* PDF.

3.1 Kompresi PDF Memakai Algoritma *Stout Code*

a. Melakukan Pembacaan Isi *File*

Nilai hexadesimal *file* PDF ini ialah 38, 32, 32, 20, 30, 20, 6F, 62, 6A, 0D, 3C, 3C, 20, 0D, 2F, 4C. Nilai data ini dimasukkan dalam tabel agar dilakukan pembacaan frekuensi. Frekuensi dibaca dengan perhitungan jumlah nilai yang sama disetiap nilai yang muncul. Pembacaan frekuensi dapat dilihat pada tabel 3 dibawah ini:

Tabel 3. Nilai *File*

Nilai	Frekuensi
20	3
32	2
0D	2
3C	2
38	1
30	1
6F	1
62	1
6A	1
2F	1
4C	1

b. Dilakukan pengurutan karakter yang memiliki frekuensi paling besar (banyak nilai yang sama) ke frekuensi paling kecil. Urutan nilai dapat dilihat dalam tabel 4 di bawah:

Tabel 4. Nilai *Bit* PDF Sample

Nilai		Bit	Frek	Bit x Frek
Hexa	Biner			
20	00100000	8	3	24
32	00110010	8	2	16
0D	00001101	8	2	16
3C	00111100	8	2	16
38	00111000	8	1	8
30	00110000	8	1	8
6F	01101111	8	1	8
62	01100010	8	1	8
6A	01101010	8	1	8
2F	00101111	8	1	8
4C	01001100	8	1	8
Total Bit				128

Dari tabel di atas, satu nilai hexadesimal (karakter) bernilai 8 bit bilangan biner. Hingga 16 bilangan *hexadesimal* memiliki nilai biner dengan jumlah 128 bit. Dalam pengubahan satuan menjadi byte maka jumlah seluruh bit dibagi 8. Maka dihasilkan 128/8 = 16.

c. Membentuk Tabel *Stout Code*

Aturan pada pembentukan kode bilangan dengan memakai *Stout Code* dapat dilihat pada bab sebelumnya. Adapun kode *Stout Code* dapat dilihat pada tabel 5 di bawah ini.

Tabel 5. Kode *Stout Code*

N	Kode *Stout Code*
1	01

2	10
3	11
4	00100
5	00101
6	00110
7	00111
8	011000
9	011001
10	011010
11	011011
12	011100

Proses berikutnya ialah dilakukan kompresi nilai pada sample dengan kode *Stout Code* yang didapat pada tabel 5 di atas. Adapun proses kompresi *file* PDF sample dapat di lihat pada tabel berikut :

Tabel 6. Kompresi Nilai *File* PDF *Sample* Dengan *Stout Code*

N	Nilai Hexa	Kode *Stout Code*	Bit	Frek	Bit x Frek
1	20	01	2	3	6
2	32	10	2	2	4
3	0D	11	2	2	4
4	3C	00100	5	2	10
5	38	00101	5	1	5
6	30	00110	5	1	5
7	6F	00111	5	1	5
8	62	011000	6	1	6
9	6A	011001	6	1	6
10	2F	011010	6	1	6
11	4C	011011	6	1	6
		Total Bit			63

Pada perhitungan tabel 6 setelah dikompresi dengan memakai *Stout Code* ialah 63 bit. Agar dilakukan pengubahan menjadi satuan byte harus dibagi 8 yaitu 63/8 = 7,8.

d. Melakukan hasil string bit *Stout Code*

Sebelum dilakukan hasil string bit *Stout Code* menjadi nilai file, terlebih dulu lakukan pemeriksaan pada panjang string bit. Pembentukan nilai bit baru hasil kompresi pada susunan nilai heksadesimal sebelum kompresi yaitu 38, 32, 32, 20, 30, 20, 6F, 62, 6A, 0D, 3C, 3C, 20, 0D, 2F, 4C(tanpa tanda koma dan spasi) menjadi nilai bit biner :

"00101101001001100100111011000011001110010000100011101101001101 1". Lalu sebelum didapat hasil. semua akhir kompresi dilakukan penambahan *string bit* yaitu *padding* serta *flag bit.* Jika sisa bagi panjang string bit terhadap 8 ialah 0 maka ditambahkan 0000001.

7 – n + "1"

7 – 7 + "1" = **1**

Bit Akhir 9 – n

Bit Akhir = 9 – 7 = 2 = **00000010**

00101101001001100100111011000011001110010000100011101101001101 1**00000010**

Total panjang bit keseluruhan setelah ada dilakukan tambah nilai bit ialah 63+1+8 =72. Lalu lakukan pemisahan bit ke beberapa kelompok. Perkelompok terdiri atas 8 bit seperti gambar di bawah

00101101 00100110 01001110 11000011 00111001 00001000 11101101 00110111 **00000010**

Berdasarkan pada pembagian kelompok nilai biner, didapatkan9 kelompok nilai biner baru (9 byte) yang telah dikompresi nilai biner penambahan bit. Lalu dilakukan pembagian, maka nilai yang telah dibagikan diubah dalam suatu karakter dengan terlebih dahulu dilakukan pencarian nilai *hexadecimal* serta *string* bit ini memakai kode ASCII dalam diketahui nilai yang sudah dikompresi. Adapun nilai yang telah terkompresi dapat di lihat dalam tabel 7 di bawah.

Tabel 7. Nilai *Hexadecimal* Terkompresi

Urutan Nilai Terkompresi	Nilai *Hexadecimal* Terkompresi
1	2D
2	26
3	4E
4	C3
5	39
6	8

7	ED
8	37
9	2

Persentase ukuran data yang dikompresi dapat dilihat pada hasil perbandingan antara ukuran data sesudah dikompresi menggunakan ukuran data sebelum dikompresi.

Ukuran data sebelum dikompresi = 128/8 = 16 byte

Ukuran data sesudah dikompresi = 72/8 = 9 byte

Dari data ini dapat dilakukan perhitungan kinerja kompresinya yaitu :

Compression Ratio (Cr)

$Cr=\frac{Ukuran\ data\ Sesudah\ Dikompresi}{Ukuran\ data\ Sebelum\ \ Dikompresi} x\ 100\%$

$Cr=\frac{9}{16} x100\%$

Cr = 56,25%

Space Saving

SS = ukuran data sebelum dikompresi – ukuran data setelah dikompresi

SS = 128- 72

SS = 56

3.2 Kompresi File PDF Menggunakan *Punctured Elias Code*

a. Mengurutkan karakter yang memiliki frekuensi terbesar (banyak nilai yang sama) ke frekuensi terkecil. Urutan nilai dapat dilihat pada tabel 8 di bawah ini

Tabel 8. Nilai *Bit* PDF Sampel

Nilai		Bit	Frek	Bit x Frek
Hexa	Biner			
20	00100000	8	3	24
32	00110010	8	2	16
0D	00001101	8	2	16
3C	00111100	8	2	16
38	00111000	8	1	8
30	00110000	8	1	8
6F	01101111	8	1	8
62	01100010	8	1	8
6A	01101010	8	1	8
2F	00101111	8	1	8
4C	01001100	8	1	8
Total Bit				128

Dari tabel 8 diatas, satu nilai hexadesimal (karakter) memiliki nilai 8 bit bilangan biner. Hingga 16 bilangan *hexadesimal* memiliki nilai biner dengan jumlah 128 bit. Dalam pengubahan satuan dalam byte maka jumlah semua bit dibagi 8. Maka didapatkan hasil 128/8 = 16.

b. Membentuk Tabel *Punctured Elias Code*

Aturan pada pembentukan kode bilangan dengan memakai *Punctured Elias Code* dapat dilihat dalam bab sebelumnya. Kode *Punctured Elias Code* dapat dilihat pada tabel 9 di bawah.

Tabel 9. Kode *Punctured Elias Code*

N	Kode P1
1	101
2	1001
3	11011
4	10001
5	110101
6	110011
7	1110111
8	100001
9	1101001
10	1100101
11	11101101

Proses berikutya ialah dilakukan kompresi nilai pada sample menggunakan kode *Punctured Elias Code* yang didapat pada tabel diatas. Proses kompresi *file* PDF sample dapat di lihat pada tabel 10 dibawah:

Tabel 10. Kompresi Nilai *File PDF Sample* Dengan *Punctured Elias Code*

N	Nilai Hexa	Kode *Stout Code*	Bit	Frek	Bit x Frek
1	20	101	3	3	9
2	32	1001	4	2	8
3	0D	11011	5	2	10
4	3C	10001	5	2	10
5	38	110101	6	1	6
6	30	110011	6	1	6
7	6F	1110111	7	1	7
8	62	100001	6	1	6
9	6A	1101001	7	1	7
10	2F	1100101	7	1	7
11	4C	11101101	8	1	8
		Total Bit			84

Dari perhitungan tabel 10 di atas setelah dikompresi memakai *Punctured Elias Code* adalah 80 bit. Untuk dilakukan pengubahan menjadi satuan byte maka dibagikan 8 yaitu 84/8 = 10, 5.

c. Melakukan hasil string bit *Punctured Elias Code*

Sebelum dilakukan hasil string bit *Punctured Elias Code* menjadi nilai *file*, terlebih dulu lakukan pemeriksaan pada panjang *string* bit. Dibentuknya nilai bit baru hasil kompresi pada susunan nilai heksadesimal sebelum kompresi yaitu 38, 32, 32, 20, 30, 20, 6F, 62, 6A, 0D, 3C, 3C, 20, 0D, 2F, 4C (tanpa tanda koma dan spasi) menjadi nilai bit biner :

"11010110011001101110011101111011110000111010011101110001100011011101111001011110110 1".

Kemudian sebelum didapatkan hasil keseluruhan akhir kompresi dilakukan penambahan *string bit* itu sendiri yaitu *padding* dan *flag bit.* Jika sisa bagi panjang string bit terhadap 8 adalah 0 maka tambahakan 0000001. Nyatakan dengan bit akhir. Sedangkan jika sisa bagi panjang *string* bit terhadap 8 adalah n(1,2,3,4,5,6,7) makan tambahkan 0 sebanyak 7-n+"1" diakhir *string bit*. Nyatakan dengan L, lalu tambahkan 0 bilangan biner dari 9 –n, nyatakan dengan bit akhir. karena jumlah string bit 74 tidak habis dibagi 8 dan sisanya 4 bit, nyatakan sisa bagi tersebut dengan nilai *n.* Maka tambahkan 0 sebanyak 7 – n + "1" akhir string bit. Nyatakan dengan L. Lalu tambahkan bilangan biner dari 9 – n. Nyatakan dengan bit akhir.

7 – n + "1"

7 – 4 + "1" = **0001**

Bit Akhir 9 – n

Bit Akhir = 9 – 1 = 7 = **00000111**

11010110011001101110011101111011110000111010011101110001100011011101111001011110110 1**000 100000111**

Total panjang bit keseluruhan setelah ada penambahan bit adalah 84+4+8 =96. Selanjutnya lakukan pemisahan bit menjadi beberapa kelompok. Setiap kelompok terdiri dari 8 bit seperti di bawah ini

11010110 01100110 11100111 01111011 11000011 10100111 01110001 10001101 11011110 01011110 1101**0001 00000111**

Berdasarkan pada pembagian kelompok nilai biner, didapatkan 12 kelompok nilai biner baru (12 byte) yang sudah terkompresi berserta nilai biner penambahan bit. Setelah pembagian dilakukan, maka nilai yang sudah dibagi diubah ke dalam suatu karakter dengan terlebih dahulu mencari nilai *hexadecimal* dan *string* bit tersebut menggunakan kode ASCII untuk mengetahui nilai yang sudah terkompresi. Adapun nilai yang sudah terkompresi dapat dilihat pada tabel 11 di bawah ini.

Tabel 11. Nilai *Hexadecimal* Terkompresi

Urutan Nilai Terkompresi	Nilai *Hexadecimal* Terkompresi
1	D6
2	66
3	E7
4	7B
5	C3
6	A7
7	71
8	8D
9	DE
10	5E
11	D1
12	7

Persentase ukuran data yang dikompresi dan dapat dilihat dari hasil perbandingan antara ukuran data setelah dikompresi dengan ukuran data sebelum dikompresi.

Ukuran data sebelum dikompresi = 128/8 = 16 byte
Ukuran data sesudah dikompresi = 96/8 = 12 byte
Berdasarkan data tersebut dapat dihitung kinerja kompresinya yaitu :
Compression Ratio (Cr)

$$Cr=\frac{Ukuran\ data\ Sesudah\ Dikompresi}{Ukuran\ data\ Sebelum\ \ Dikompresi} x\ 100\%$$

$$Cr=\frac{12}{16} x100\%$$

$$Cr = 75\%$$

Space Saving
SS = ukuran data sebelum dikompresi – ukuran data setelah dikompresi
SS = 128- 96s
SS = 32

3.3 Hasil Perbandingan Algoritma *Punctured Elias Code dan Stout Code*

Perbandingan algortima *Punctured Elias Code dan Stout Code* ialah membandingan rasio kompresi dan *space saving* dari kedua algoritma. Dimana setelah dilakukan perhitungan dengan algoritma *Punctured Elias Code dan Stout Code* maka hasil dari rasio kompresi dan *space saving* didapatkan. Hasil dari rasio kompresi algoritma *Stout Code* sebesar 56,25% dan hasil rasio dari algoritma *Punctured Elias Code* sebesar 75%. Dari hasil rasio kedua algoritma dapat diketahui bahwa algoritma *Punctured Elias Code* lebih akurat dalam melakukan kompresi file dokumen PDF.

4. KESIMPULAN

Hasil kesimpulan dalam penelitian ini dimana kompresi *file* pdf dapat dilakukan dengan mengubah kedalam bentuk heksadesimal untuk dapat melakukan perhitungan. Bilangan heksadesimal yang didapat diolah menggunakan algoritma *stout code* dan *punctured elias code* sehingga menghasilkan nilai yang baru yang akan merubah ukuran *file* pdf. Algoritma *stout code* dan *punctured elias code* dapat dilakukan perbandingan setelah dilakukan perhitungan. Dengan skala perbandingan yaitu rasio kompresi dan *space saving*. Dengan perbandingan yang dilakukan maka diketahui bahwa persentase rasio kompresi algoritma *punctured elias code* lebih besar daripada algoritma *stout code*. Rasio kompresi dan space dapat diketahui dengan melakukan mengubah nilai awal *file* pdf dengan nilai heksadesmial. Didapatkan nilai rasio kompresi algoritma *stout code* sebesar 56,25% dan hasil rasio kompresi dengan algoritma *punctured elias code* sebesar 75%.

REFERENCES

[1] David Salomon, G. M. (2010). *Handbook Of Data Compression*. Spinger.
[2] Kompresi, A., & Codes, A. S. (2021). *Perancangan Aplikasi Kompresi File Teks Menggunakan Algoritma Stout Codes*. *9*, 183–188.
[3] Lamsah, & Utomo, D. P. (2020). Penerapan Algoritma Stout Codes Untuk Kompresi Record Pada Databade Di Aplikasi Kumpulan Novel. *KOMIK (Konferensi Nasional Teknologi Informasi Dan Komputer)*, *4*(1), 311–314. https://doi.org/10.30865/komik.v4i1.2710
[4] Putra, D. (2010). *Pengolahan Citra Digital*. C.V Andi Offset.
[5] Riyansyah, D. (2019). Perancangan Aplikasi Kompresi File Video Menggunakan Algoritma Interpolative Coding. *KOMIK (Konferensi Nasional Teknologi Informasi Dan Komputer)*, *3*(1), 392–397. https://doi.org/10.30865/komik.v3i1.1618
[6] Yansyah, D. A., & Pendahuluan, I. (2015). *PERBANDINGAN METODE PUNCTURED ELIAS CODE DAN*. *2*(6), 33–36.
[7] Yuni, D. (2020). Kompresi File Aplikasi Hadist Dengan Menggunakan Algoritma Punctured Elias Code. *KOMIK (Konferensi Nasional Teknologi Informasi Dan Komputer)*, *4*(1), 364–367. https://doi.org/10.30865/komik.v4i1.2721
[8] S. D. Nasution, "Data Compression Using *Stout codes*," *Ijics*, vol. 3, no. 1, pp. 28–33, 2019.
[9] N. Suryana and D. Rosiyadi, "The Reduction Method of ISO image for Operating System of Open Source Based on Linux Metode Reduksi ISO Image Sistem Operasi Open Source Berbasis Linux," vol. 6, no. 2, 2012.
[10] 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.
[11] C. T. Utari, P. Studi, M. Teknik, U. S. Utara, and K. Citra, "IMPLEMENTASI ALGORITMA RUN LENGTH ENCODING UNTUK PERANCANGANAPLIKASI KOMPRESI DAN DEKOMPRESI," vol. V, no. 2, pp. 24–31, 2016.
[12] 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.