Bulletin of Information System Research (BIOS)
Volume 2, No 1, December 2023 Page: 10-20
ISSN 2963-2455 (media online)
https://journal.grahamitra.id/index.php/bios

Perbandingan Algoritma Stout Code Dan Punctured Elias Code Dalam Mengkomprsi File Matroska Video Container

Muhamad Rasidi

Ilmu Komputer dan Teknologi Informasi, Teknik Informatika, Universitas Budi Darma, Medan, Indonesia
Email: muhamadrasidi@gmail.com

Abstrak−Ukuran file video tergolong sangat besar sehingga megakibatkan perlunya penyimpanan yang cukup besar dan file video dengan durasi yang panjang sangat mempengaruhi ruang penyimpanan dan waktu pengiriman yang cukup lama sehingga perlu dilakukannya kompresi data. Kompresi file ini bertujuan untuk memperkecil ukuran file video agar menghemat ruang penyimpanan dan waktu pengiriman file menjadi lebih cepat. Didalam kompresi banyak algoritma untuk melakukan proses kompresi. Diantaranya stout code dan punctured elias code. Dengan terdapatnya beberapa algoritma tersebut pelu dilakukan pemilihan algoritma yang lebih tepat sehingga perlu dilakukan perbandingan algoritma kompresi. Setelah dilakukan perhitungan dengan algoritma stout code dan punctured elias 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 video MKV.

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

Abstract−The size of the video file is very large so that it requires a large enough storage and video files with a long duration greatly affect the storage space and the long delivery time so that data compression is necessary. This file compression aims to reduce the size of video files in order to save storage space and make file transfer times faster. In compression there are many algorithms to perform the compression process. Among them stout code and punctured elias code. With the existence of several algorithms, it is necessary to select a more appropriate algorithm so that it is necessary to compare the compression algorithms. After carrying out calculations using the stout code and punctured Elias code algorithms, the results of the compression ratio and space saving are obtained. The compression ratio result of the stout code algorithm is 56.25% and the ratio result of the punctured Elias code algorithm is 75%. From the ratio results of the two algorithms, it can be seen that the punctured Elias code algorithm is more accurate in compressing MKV video files.

Keywords: Compression; Video Files; Comparison; Stout Code; Punctured Elias 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 video tergolong sangat besar dan terjadi pengaruh pada ruang penyimpanan serta waktu berkirim data juga tergolong lama dengan ukuran file yang begitu besar. Oleh karenanya perlunya dilakukan kompresi dalam dimampatkannya isi file video (.mkv). Ketika proses kompresi dilakukan harus adanya algoritma pendukung dalam kompresi. Algoritma-algoritma memiliki kelebihan serta kekurangan saat melakukan proses kompresi data.

Matroska Kontainer Multimedia atau yang sering juga disebut dengan matroska video container adalah salah satu format kontainer multimedia atau biasa disebut dengan .mkv. Wadah mkv dapat menggabungkan audio, video, dan subtitle ke dalam satu file, bahkan jika elemen-elemen tersebut menggunakan berbagai jenis penyandian.

Pada penelitian kali dilakukan perbandingan algoritma untuk mengetahui algoritma manakah yang akurat dalam proses komperesi file berfomat .mkv. Algoritma yang akan dibandingkan yaitu algoritma stout code dan punctured elias code. Kedua algoritma tersebut mereka algoritma untuk proses kompresi. Variabel perbadingan kedua algoritma adalah rasio kompresi dan space saving.

Punctured adalah sebuah bilangan integer yang dirancang Peter Fenwick 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, agar dipersingkatnya rangkaian kode itu, 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 variable dalam bilangan bulat, mirip dengan serta lebih umum pada Elias omega serta Even – Rodeh code. Stout code memiliki sifat universal serta asimtotik optimal sebelumnya pembaca diinganken bawah panjangn L dari Bilangan bulat n diberikan oleh L =1 + log2 n.

Dalam penelitian yang lalu dilakukan oleh Desvika Riyansyah dengan penelitian yang menerapkan Algoritma Interpolative Coding dalam Perancangan sebuah Aplikasi Kompresi File Video sehingga menyimpulkan bahwa dengan penerapan algoritma interpolative coding telah mendapatkan rasio kompresi sebesar 47%[2].

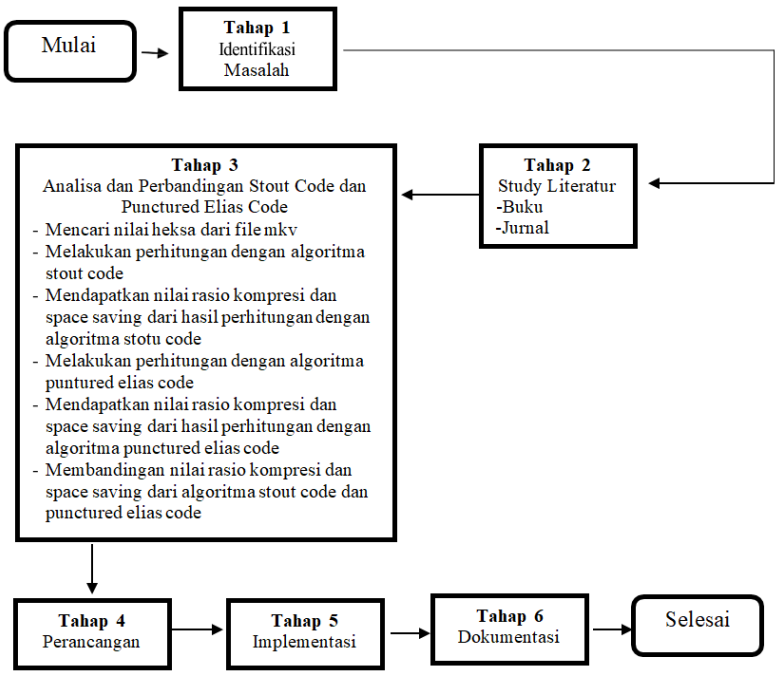
Bulletin of Information System Research (BIOS)
Volume 2, No 1, December 2023 Page: 10-20
ISSN 2963-2455 (media online)
https://journal.grahamitra.id/index.php/bios

Penelitian lainnya yang dilakukan oleh Apujudin dalam skripsinya yaitu pemanfaatan Algoritma Stout Code dalam Perancangan Aplikasi Kompresi File Teks, dimana ia 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 bahasan penelitian dalam mengkompresi file text dilakukan terhadap dua metode yang akan di bandingkan hasil akhirnya yaitu metode Punctured Elias Code dengan metode Huffman sehingga 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 bahasan Memperkecil Ukuran File Hadist dengan Algoritma Canggih 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 membahas Optimalisasi Penyimpanan Data Novel dengan Kompresi Record Database Berbasis Stout Code sehingga 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. Dibawah ini merupakan alur sederhana proses pengumpulan data dari penelitian yang dilakukan hingga selesaikan penelitian ini.



Gambar 1. Tahapan Penelitian

1. 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 stout code dan punctured elias code dalam melakukan kompresi.
2. Tahap Study Literarut
   Pada penelitian pengumpulan data yang berkaitan dengan algoritma stout code dan punctured elias code. Penulis mengumpulkan sebagai referensi baik dari buku, jurnal dan situs internet yang memiliki keterkatian dengan penelitian.
3. Analisa
   Tahapan analisa digunakan untuk mengetahui apa yang menjadi sumber masalah pada perbandingan algoritma stout code dan puntured elias code, sehingga penyelesaian yang dihasilkan nantinya dapat mengatasi permasalahan yang ada.

4. Perancangan
   Tahap ini merupakan perancangan serta pengujian terhadap sistem yang telah dibangun sehingga dalam pengujian tersebut dapat diketahui apakah sistem telah bekerja dengan benar dalam perbandingan proses kompresi dan dekompresi ukuran file mkv.
5. Implementasi
   Pada tahap ini dilakukan pengimplementasian file mkv menggunakan algoritma stout code dan punctured elias code.
6. Dokumentasi
   Tahap dilakukan dengan pendokumentasian hasil analisa dan pengujian secara tertulis dalam bentuk laporan skripsi.

2.2 Kompresi

Kompresi ailah 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 dihasilkannya ukuran data jadi lebih kecil tetapi kualitas informasi tetap terjaga[3]. Kompresi file video ialah cara pengecilian jumlah tiap bit yang direpresentasikan dalam file video dengan data yang jadi lebih kecil. Kompresi data memiliki arti teknik pememampatan data sehinnga didapatkan ukuran data lebih kecil dari ukuran awal serta menjadi efisien dalam menyimpan ataupun waktu menukar data lebih singkat[3]. Teknik kompresi di klarifikasi dalam 2 (Dua) berdasarkan hasil kompresi[4], antara lain[7][8][9]:

1. Lossless Compression
   Tidak ada kehilangan informasi pada teknik lossless compression. Apabila data dilakukan pemanpatan dengan lossless, data asli dapat kembali sama dari data yang sudah dilakukan pemampatan, dalam artian data asli tetap sama sebelum serta sesudah dimampatkan[4]. Secara umum teknik lossless dipakai pada penerapan yang tak dapat mentoleransi setiap hal berbeda antar data asli serta data yang sudah mengalami rekonstruksi. Data memiliki bentuk tulisan misalnya file, harus dimampatkan memakai teknik lossys, sebab hilangya sebuah karakter dapat terjadinya kesalapahaman..
2. Lossy Compression
   Cari ini ialah hilangnya beberapa informasi. File yang dimampatkan menggunakan cara ini secara umum tak dapat terekonstruksi sama persis dari data asli[4]. Pada banyak penerapan, rekonstruksi tepat tidak menjadi masalah.

Terdapat beberapa teknik yang menjadi parameter dalam penunjukkan kualitas ataupun kinerja pada suatu cara kompresi[5], yaitu:

1. Compression Ratio (CR)
   Compression Ratio (CR) ialah persentase perbandingan antara data yang sudah dikompresi dengan data yang belum dikompresi. Secara sistematis dapat dituliskan sebagai berikut:

$$CR = \frac{Ukuran\ data\ setelah\ di\ kompresi}{Ukuraan\ data\ sebelum\ di\ kompreis} * 100\% \quad (1)$$

2. Redudancy (Rd)
   Redudancy ialah hasil penilaian nilai rasio dengan hasil nilai rasio kompresi. Secara sistematis dapat dituliskan sebagai berikut :

$$Rd = \frac{File\ sebelum\ di\ kompresi - File\ setelah\ di\ kompresi}{Ukuraan\ file\ sebelum\ di\ kompresi} * 100\% \quad (2)$$

2.3 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*[10]. Langkah dalam dibangunnya kode *Punctured Elias Codes* ialah sebagai berikut[11][12]:

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

N	Binary Of n	Reserved	Flag	Flag | Reserved	P1
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

Pada makalah pendek Quentin stout diperkenalkan dua keluarga RI serta SI pada rekursi, kode panjang variable dalam bilangan bulat, sama deangan serta lebih umum pada Elias omega serta Even – Rodeh code. Stout code memiliki sifat universal serta asimtotik optimal sebelum dilakukan pembacaaan. Sebelum dilakukan pembacaaan diinganken bawah panjang L dari Bilangan bulat n diberi oleh L =1 + log2 n (persmaan(2.1))[13][14].

Dua kelompok kode tergantung dalam pilihan parameter interger 1 sekali a nilai(lebih besar dari ataupun sama dengan 2) dalam 1 telah dilakukan pemilihan kata sandi pada keluarga pertama terdiri atas awalan RI (n) serta diakhirit 0n akhiran ialah nilai biner pada n pada bit L didahulukan oleh permisa 0 tunggal.itu awalan RI (n) terdiri atas grup panjang mulai dengan panjang L dari n yaitu didahului panjangn L1 serta L maka panjang L2 dari L1 serta selanjutnya hingga panjang Li mencapai yang cukup pendet dalam masuk kelompok 1- bit. Perhatikan jika kelompok panjang (kecuali mungkin yang paling kiri) dimulai pada 1.dengan demikian permisa 0 tunggal ditunjukkan akhir kelompok panjang[5][15].

Contohnya apabila melakukan pemilihan 1 = 2 serta dikodekan interger 985 – bit n yang memiliki nilia tepat takrelevan. Akhiran ialah string 986 – bit 0n serta awal mulai atas grup L = 985 =11110110012 panjang L1 dari grup ini ialah 1010 =10102 panjang L2 pada kelompok kedua ialah 4 = 1002 serta panjang L3 ialah 3 = 112 lengkap. oleh karenya codeword iaah 11|100|1010|1111011001|0|n perhatikan jika kelompok panjang mulai dari 1, yang disirahtkan jika semua kelompok diikuti 1, kecuali kelompok panjang akhir yang diikuti 0. Penguraian sederhana. Dekoder mulai dengan membaca bit pertama. Ini ialah panjang grup selanjutnya. Semakin banyak grup panjang dibaca, hingga grup ditemukan yang diikuti oleh 0. Ini menunjukkan jika grup selanjutnya ialah n itu sendiri. Dengan latar belakang tersebut, definisi rekursif pada awalan Rl mudah dilakukan pembacaan serta mudah dipahami. Dipakainya notasi L = 1 +? Log2 n? serta memiliki lambang B (n, l) representasi biner l-bit (kode beta) pada integer n. Jadi, B (12, 5) = 01100. Untuk l ≥ 2, awalan didefinisikan oleh :

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

Mereka yang akrab dengan Even – Rodeh code mungkin sudah menyadari bahwa kode ini identik dengan R3. Lebih lanjut, kode omega Elias (Bagian 3.4) berada di antara R2 dan R3 dengan dua perbedaan: (1) kode omega mengkodekan jumlah Li − 1 dan (2) pemisah 0 ditempatkan di sebelah kanan n.

$R_2(985)$=	11 100 1010 1111011001	$R_2(31,925)$=	11 100 1111 111110010110110
$R_2(985)$=	100 1010 1111011001	$R_2(31,925)$=	100 1111 111110010110110
$R_2(985)$=	1010 1111011001	$R_2(31,925)$=	1111 111110010110110
$R_2(985)$=	01010 1111011001	$R_2(31,925)$=	01111 111110010110110
$R_2(985)$=	001010 1111011001	$R_2(31,925)$=	001111 111110010110110

Tabel pendek mencantumkan awalan Rl (n) untuk 2 ≤ l ≤ 6 dan 985-bit dan 31.925- bit integer. Jelas bahwa awalan terpendek diperoleh untuk nilai parameter l = L = 1+? Log2 n ?. Nilai l yang lebih besar menghasilkan awalan yang sedikit lebih lama, sedangkan nilai yang lebih kecil membutuhkan kelompok yang lebih panjang dan karenanya menghasilkan awalan yang jauh lebih lama. Manipulasi aljabar dasar menunjukkan bahwa rentang panjang terbaik L untuk parameter yang diberikan l diberikan oleh [2s, 2e - 1] di mana s = 2l − 1 dan e = 2l - 1 = 2s - 1 (ini adalah rentang panjang terbaik L, bukan bilangan bulat terbaik n). Tabel berikut mencantumkan interval ini untuk beberapa nilai l dan menjelaskan bahwa parameter kecil, seperti 2 dan 3, cukup untuk sebagian besar aplikasi praktis kompresi data. Parameter l = 2 terbaik untuk bilangan bulat yang panjangnya 2 hingga 7 bit, sedangkan l = 3 yang terbaik untuk bilangan bulat yang panjangnya 8 hingga 127 bit[5].

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

Bulletin of Information System Research (BIOS)
Volume 2, No 1, December 2023 Page: 10-20
ISSN 2963-2455 (media online)
https://journal.grahamitra.id/index.php/bios

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:

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

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
8	01 1000	000 1000
15	01 1111	000 1111
16	10 10000	001 10000
32	11 100000	010 100000
64	00 100 1000000	011 1000000
128	00 101 10000000	100 10000000
256	00 110 100000000	101 100000000
512	00 111 1000000000	110 1000000000
1024	01 1000 10000000000	111 10000000000
2048	01 1001 100000000000	000 1000 100000000000

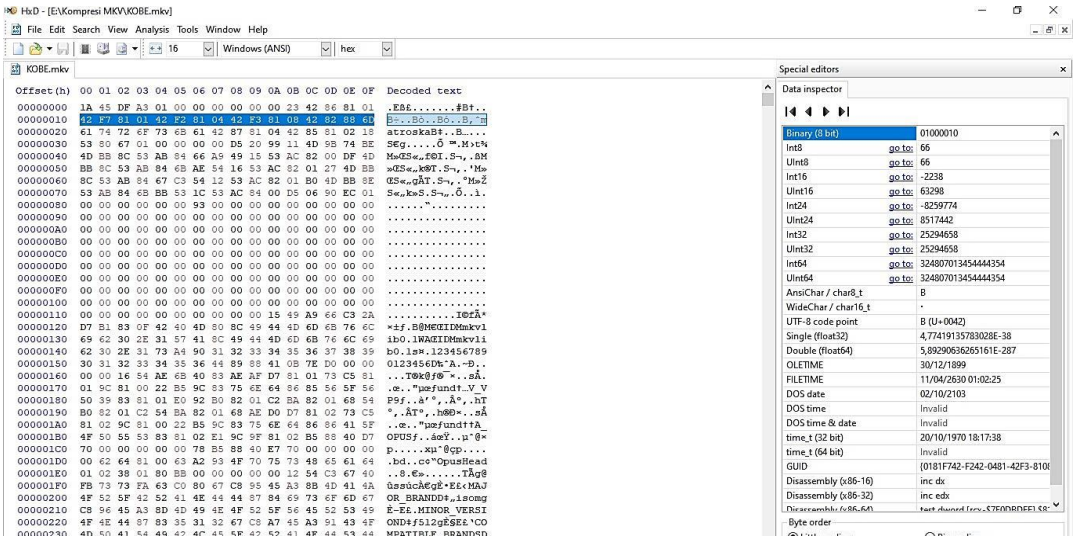
Awalan S2 (64), misalnya, dimulai dengan grup 7-bit 1000000 = 64 dan bergantung padanya S2 (7−1−2) = S2 (4) = 00 | 100. Kami menekankan lagi bahwa tabel hanya mencantumkan awalan, bukan codeword lengkap. Setelah ini dipahami, tidak sulit untuk melakukannya lihat bahwa kode Stout kedua adalah kode awalan. Setelah codeword diberikan, itu tidak akan menjadi awalan dari codeword lainnya[5]. Jadi, misalnya, awalan dari semua codewords untuk bilangan bulat 64-bit dimulai dengan awalan 00 100 dari bilangan bulat 4-bit, tetapi setiap codeword untuk bilangan bulat 4-bit memiliki 0 mengikuti 00 100, sedangkan codewords untuk bilangan bulat 64-bit memiliki 1 mengikuti 00 100.

3. HASIL DAN PEMBAHASAN

File video yang berekstensi MKV mempunyai ukuran yang cukup besar, semakin lama waktu pada file video maka makin besar pula ruang penyimpanan yang diperlukan, dan proses pengiriman file tergolong cukup lama. Saat dilakukannya proses kompresi file MKV harus dilakukan terlebih dahulu analisa pada file MKV. Kompresi *file* video dengan ekstensi MKV memiliki ukuran yang sangat besar. Dengan dilakukannya kompresi *file* MKV, file yang memiliki ukuran lebih besar akan dikompres agar ukuran lebih kecil serta menghemat ruang penyimpanan serta diketahui algoritma yang lebih akurat dalam melakukan kompresi *file* MKV dengan algoritma *stout code* dan *punctured elias code*. Pada penelitian ini, akan dibahas 2 proses utama ialah proses kompresi serta proses dekompresi, dan peneliti melakukan kompresi menggunakan dua algoritma yaitu algoritma *stout code* dan *punctured elias code*. Sebelum *file* MKV dikompres, lebih dulu dilakukan pembacaan *file* MKV agar didapatkan nilai hexadesimal menggunakan aplikasi HxD. Berikut contoh *file* MKV yang dikompres serta didekompresi.

Tabel 3. Sampel File MKV yang akan dikompresi

Nama File	Kobe
Extension File	MKV
Size	13.3 Mb



Gambar 2. Isi File MKV dari Hasil HxD

Dari ambar diatas didapat nilai hexadesimal *file* MKV sampel. Untuk keperluan perhitungan manual, maka hanya akan diambil sampel nilai sebanyak 16 karakter nilai *hexadesimal file* MKV sampel. Nilai *hexadesimal* diambil dari sisi kiri ke kanan yaitu 42, F7, 81, 01, 42, F2, 81, 04, 42, F3, 81, 08, 42, 82, 88, 6D. 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 dibawah ini:

Tabel 4. Nilai *File*

Nilai	Frekuensi
42	4
F7	1
81	3
01	1
F2	1
04	1
F3	1
08	1
82	1
88	1
6D	1

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

Tabel 5. Nilai *Bit* MKV Sample

Nilai		Bit	Frek	Bit x Frek
Hexa	Biner			
42	01000010	8	4	32
81	10000001	8	3	24
F7	11110111	8	1	8
01	00000001	8	1	8
F2	11110010	8	1	8
04	00000100	8	1	8
F3	11110011	8	1	8
08	00001000	8	1	8
82	10000010	8	1	8
88	10001000	8	1	8
6D	01101101	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 182/8 = 16.

Bulletin of Information System Research (BIOS)
Volume 2, No 1, December 2023 Page: 10-20
ISSN 2963-2455 (media online)
https://journal.grahamitra.id/index.php/bios

3.1 Analisa Proses Kompresi File MKV Memakai Algoritma *Stout Code*

Aturan pada pembentukan kode bilangan dengan memakai *stout code* dapat dilihat pada bab sebelumnya. Adapun kode *stout code* dapat dilihat pada tabel di bawah ini.

Tabel 6. 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 6 di atas. Adapun proses kompresi *file* MKV sample dapat di lihat pada tabel berikut :

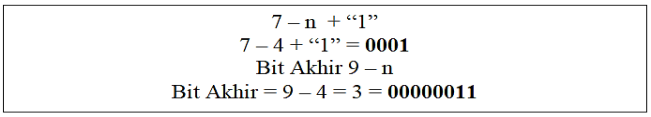
Tabel 7. Kompresi Nilai *File* MKV *Sample* Dengan *Stout Code*

N	Nilai Hexa	Kode Stout Code	Bit	Frek	Bit x Frek
1	42	01	2	4	8
2	81	10	2	3	6
3	F7	11	2	1	2
4	01	00100	5	1	5
5	F2	00101	5	1	5
6	04	00110	5	1	5
7	F3	00111	5	1	5
8	08	011000	6	1	6
9	82	011001	6	1	6
10	88	011010	6	1	6
11	6D	011011	6	1	6
		Total Bit			60

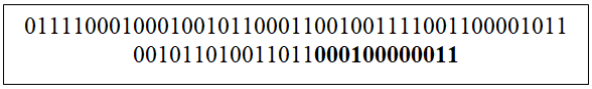
Pada perhitungan tabel di atas setelah dikompresi dengan memakai *stout code* ialah 60 bit. Agar dilakukan pengubahan menjadi satuan byte harus dibagi 8 yaitu 60/8 = 7,5. Sebelum dilakukan hasil string bit stout code menjadi nilai file, terlebih dulu lakukan pemeriksaan pada panjang string bit. Pembentukan nilai bit baru hasil kompresi pada susunan nilai heksadesimal sebelum kompresi yaitu 42, F7, 81, 01, 42, F2, 81, 04, 42, F3, 81, 08, 42, 82, 88, 6D (tanpa tanda koma dan spasi) menjadi nilai bit biner :
"011110001000100101100011001001111001100001011001011010011011".

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. Dinyatakan dengan bit akhir. Sedangkan jika sisa bagi panjang *string* bit terhadap 8 adalah n(1,2,3,4,5,6,7) maka 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 60 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.



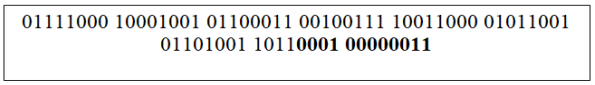
Gambar 3. Perhitungan Penambahan Bit



Gambar 4. String Bit yang telah dilakukan penambahan

Bulletin of Information System Research (BIOS)
Volume 2, No 1, December 2023 Page: 10-20
ISSN 2963-2455 (media online)
https://journal.grahamitra.id/index.php/bios

Total panjang bit keseluruhan setelah ada dilakukan tambah nilai bit ialah 60+4+8 =73. Lalu lakukan pemisahan bit ke beberapa kelompok. Perkelompok terdiri atas 8 bit seperti gambar di bawah



Gambar 5. Pembagian *String* bit

Berdasarkan pada pembagian kelompok nilai biner, didapatkan 9 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 di bawah.

Tabel 8. Nilai *Hexadecimal* Terkompresi

Urutan Nilai Terkompresi	Nilai *Hexadecimal* Terkompresi
1	78
2	89
3	63
4	27
5	98
6	59
7	69
8	B1
9	3

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 :

1. *Compression Ratio* (CR)

   $$CR = \frac{Ukuran\ data\ setelah\ di\ kompresi}{Ukuraan\ data\ sebelum\ di\ kompreis} * 100\%$$

   $$CR = \frac{9}{16} * 100\%$$

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

2. *Space Saving* (Ss)

   $$S_S = 100\% - C_R$$

   $$S_S = 100\% - 56{,}25\%$$

   $$S_S = 43{,}75\%$$

3.2 Analisa Proses Kompresi File MKV Menggunakan 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 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

Bulletin of Information System Research (BIOS)
Volume 2, No 1, December 2023 Page: 10-20
ISSN 2963-2455 (media online)
https://journal.grahamitra.id/index.php/bios

Proses berikutnya ialah dilakukan kompresi nilai pada sample menggunakan kode *punctured elias code* yang didapat pada tabel diatas. Proses kompresi *file* MKV sample dapat di lihat pada tabel dibawah:

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

N	Nilai Hexa	Kode Stout Code	Bit	Frek	Bit x Frek
1	42	101	3	4	12
2	81	1001	4	3	12
3	F7	11011	5	1	5
4	01	10001	5	1	5
5	F2	110101	6	1	6
6	04	110011	6	1	6
7	F3	1110111	7	1	7
8	08	100001	6	1	6
9	82	1101001	7	1	7
10	88	1100101	7	1	7
11	6D	11101101	8	1	8
		Total Bit			81

Dari perhitungan tabel di atas setelah dikompresi memakai *punctured elias code* adalah 80 bit. Untuk dilakukan pengubahan menjadi satuan byte maka dibagikan 8 yaitu 80/8 = 10, 125. 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 42, F7, 81, 01, 42, F2, 81, 04, 42, F3, 81, 08, 42, 82, 88, 6D (tanpa tanda koma dan spasi) menjadi nilai bit biner :
"10111011100110001101110101100111001110111101111001100001101110100111001011110 1101".

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 2 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 – 1 + "1" = **0000001**
Bit Akhir 9 – n
Bit Akhir = 9 – 1 = 7 = **00000111**

Gambar 6. Perhitungan Penambahan bit

10111011100110001101110101100111001110111101111011
1100110000110111010011100101111011010**00000100**
000111

Gambar 7. *String* bit yang telah dilakukan penambahan

Total panjang bit keseluruhan setelah ada penambahan bit adalah 81+7+8 =88. Selanjutnya lakukan pemisahan bit menjadi beberapa kelompok. Setiap kelompok terdiri dari 8 bit seperti gambar di bawah ini

10111011 10011000 11011101 01100111 00111011 11011110
01100001 10111010 01110010 11110110 1**0000001 00000111**

Gambar 8. Pembagian *String* bit

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 di bawah ini.

Tabel 4.11 Nilai *Hexadecimal* Terkompresi

Urutan Nilai Terkompresi	Nilai *Hexadecimal* Terkompresi
1	BB

Urutan Nilai Terkompresi	Nilai *Hexadecimal* Terkompresi
2	98
3	DD
4	67
5	3B
6	DE
7	61
8	BA
9	72
10	F6
11	81
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 :

1. *Compression Ratio* (CR)

   $$CR = \frac{Ukuran\ data\ setelah\ di\ kompresi}{Ukuraan\ data\ sebelum\ di\ kompreis} * 100\%$$

   $$CR = \frac{12}{16} * 100\%$$

   $$CR = 75\%$$

2. *Space Saving* (Ss)

   $$S_S = 100\% - C_R$$

   $$S_S = 100\% - 75\%$$

   $$S_S = 25\%$$

3.3 Perbandingan Algoritma Stout Code dan Punctured Elias Code

Setelah dilakukan perhitungan dengan algoritma *stout code* dan *punctured elias 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 video MKV. Adapun tabel perbandingan dapat di lihat pada tabel di bawah ini.

Tabel 12. Nilai Perbandingan

Stout Code	Punctured Elias Code
Rasio Kompresi 56,25%	Rasio Kompresi 75%

4. KESIMPULAN

Kesimpulan yang dapat ditarik setalah dilakukan penngerjaan pada bab-bab sebelumnya yaitu kompresi file matroska video container 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 matroska video container. 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 matroska video container 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] Allba Firdaus Oki dan Nova Sasmita, 2018. “Sistem Pendukung Keputusan pemilihan Jurusan Menggunakan Metode Profile matching pada SMK Negeri 2 Sekayu”. Skripsi. Palembang : Sekolah Tinggi Manajemen Informatika Dan Komputer Palcomtech

[2] Winarsih Juli Ayu, 2017. "Sistem Pendukung Keputusan Penilaian Kinerja Karyawan untuk Kenaikan Jabatan pada PT SMS Cengkareng Barat dengan Metode Profile Matching". Skripsi. Jakarta : STMIK Nusa Mandiri Jakarta.
[3] Diaz Nickolas dan Sulindawati, 2020. "Sistem Pendukung Keputusan Seleksi Calon Peserta Paskibraka Kab. Karo Menggunakan Profile Matching". Jurnal Teknik Informatika. Vol. 1, No. 2, Desember 2020, hlm. 87-91. https://doi.org/10.20884/1.jutif.2020.1.2.28.
[4] "Pengertian Sistem Pendukung Keputusan (SPK) Menurut Para Ahli | Kumpulan Pengertian." https://www.kumpulanpengertian.com/2015/04/pengertian-sistem- pendukung-keputusan.html (accessed Sep. 16, 2021).
[5] Ninla Elmawati Falabiba, "**済無**No Title No Title No Title," pp. 7–26, 2019.
[6] "Pemandu sorak - Wikipedia bahasa Indonesia, ensiklopedia bebas." https://id.wikipedia.org/wiki/Pemandu_sorak (accessed Sep. 16, 2021).
[7] R. Fadillah, A. S. Idris, D. Marlina, G. Lubis, and R. Meisisri, "Implementasi Algoritma Fast Encryption Algorithm (FEAL) Dan Algoritma Fibonacci Mengamankan File Teks," pp. 295–300, 2022.
[8] B. W. Transform and A. S. Harahap, "Analisis Dan Implementasi Kompresi File Citra Menggunakan Algoritma," vol. 1, no. 1, pp. 6–12, 2021.
[9] E. Prayoga and K. M. Suryaningrum, "IMPLEMENTASI ALGORITMA HUFFMAN DAN RUN LENGTH ENCODING PADA APLIKASI KOMPRESI BERBASIS WEB," J. Ilm. Teknol. Infomasi Terap., vol. 4, no. 2, pp. 92–101, 2018, doi: 10.33197/jitter.vol4.iss2.2018.154.
[10] B. Sari, "Perbandingan Metode Profile Matching Dan Simple Additive Weighting Pada Penentuan Jurusan Siswa Kelas X Sma N 2 Ngaglik," *Data Manaj. dan Teknol. Inf.*, vol. 16, no. 1, p. 16, 2015.
[11] S. M. Hardi, M. R. Putra, J. T. Tarigan, and I. Jaya, "Comparison of Text Data Compression Using Yamamoto Recursive Codes and Punctured Elias Codes," in Journal of Physics: Conference Series, 2020, vol. 1566, no. 1, p. 12083.
[12] M. Mawar, "Perancangan Aplikasi Kompresi File Pdf Dengan Menerapkan Algoritma Punctured Elias Codes," Inf. dan Teknol. Ilm., vol. 7, no. 3, pp. 217–223, 2020.
[13] 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. 1209–1215, 2020.
[14] R. T. A. Sitanggang, "Penerapan Algoritma Stout Codes Untuk Kompresi Databases Pada Pinjaman Online," Bull. Inf. Technol., vol. 3, no. 3, pp. 189–194, 2022.
[15] [H. S. Purba, "Implementasi Algoritma Stout code untuk Kompresi Record Database pada Website Dinas Pariwista dan Kebudayaan Kabupaten Dairi," Pelita Inform. Inf. dan Inform., vol. 10, no. 4, pp. 129–133, 2022.