BIOSTech: Bulletin of Computer Science and Information Technology
Vol 1, No 1, Desember 2022
Hal: 22-29
Available Online at https://journal.grahamitra.id/index.php/biostech

Implementasi Algoritma Prefix Codes untuk Kompresi File Video Hasil Ekstra Aplikasi Kinemaster

Agustrina Sihotang

Fakultas Ilmu Komputer dan Teknologi Informasi, Program Studi Teknik Informatika, Universitas Budi Darma, Medan, Indonesia
Email: rinasihotang95@gmail.com

Abstrak−*Kinemaster* merupakan aplikasi editor video profesional yang sangat lengkap yang dapat mengedit berbagai lapisan video, gambar dan juga teks yang dilengkapi dengan pemotongan video yang tepat. Ketika pengguna melakukan aktivitas mengedit video, dan menyimpan *file* video hasil edit tersebut, maka akan menghasilkan *file video* yang memiliki resolusi yang tinggi. Oleh karena semakin tinggi resolusi dan durasi dari suatu video maka semakin besar pula ukuran *file* video tersebut. Oleh karena itu ukuran *file* tersebut akan menimbulkan masalah karena dibatasi dengan kapasitas penyimpanan. Untuk mengatasi masalah tersebut maka didapatkan solusi dengan melakukan kompresi. Algoritma *Prefix Codes* merupakan teknik kompresi *lossless*, yang tidak menghilangkan informasi sebelumnya, dimana hasil dekompresi dari video yang terkompresi sama dengan *file* video aslinya. Dengan menggunakan algoritma *Prefix Codes* menghasilkan kinerja *Ratio of Compression* sebesar 2 *bit*, *Compression Ratio* sebesar 50% dan *Space Savings* sebesar 50% . Hasil tersebut membuktikan bahwa kompresi *file* video hasil ekstra *Kinemaster* dengan menerapkan algortitma *Prefix Codes* dapat memperkecil ukuran *file* video yang besar menjadi lebih kecil dari ukuran sebelumnya.

Kata Kunci: Kompresi; File Video; Kinemaster; Prefix Codes.

Abstract−Kinemaster is a very complete professional video editor application that can edit multiple layers of video, images and text, complete with precise video cropping. When the user performs video editing activities, and saves the edited video file, it will produce a video file that has a high resolution. Because the higher the resolution and duration of a video, the greater the video file size. Therefore the file size will cause problems because it is limited by storage capacity. To solve this problem, a solution is obtained by compression. The Prefix Codes algorithm is a lossless compression technique, which does not remove previous information, where the decompression of the compressed video is the same as the original video file. By using the Prefix Codes algorithm, the performance is a Ratio of Compression of 2 bits, a Compression Ratio of 50% and a Space Savings of 50%. These results prove that the compression of the extra video files by Kinemaster by applying the Prefix Codes algorithm can reduce the size of large video files to be smaller than the previous size.

Keywords: Compresi; Video Files; Kinemaster; Prefix Codes.

1. PENDAHULUAN

Seiring dengan perkembangan zaman yang semakin maju video pun semakin banyak manfaatnya bagi masyarakat. Dengan adanya video semua orang akan mendapatkan informasi lebih jelas, dapat menghibur atau menambah pengetahuan manusia dan membantu manusia berkomunikasi jarak jauh. Dalam penyajian sebuah video yang menarik dibutuhkan suatu aplikasi untuk mengedit video. Aplikasi untuk mengedit video sangat banyak salah satunya adalah *Kinemaster*. *Kinemaster* merupakan aplikasi yang sangat kuat untuk mengedit video dengan fitur untuk pengembangan video penuh. Terdapat beberapa format dalam *file* video seperti MP4, MPEG, AVI, MKV dan lain sebagainya. Format yang sering digunakan dalam penyimpanan *file* video adalah MP4, karena sangat mudah diputar ke beberapa aplikasi pemutar video.

Dalam mengedit *file* video ada beberapa permasalahan yang ditemukan saat selesai mengedit terutama hasil unduhan dari *file* video tersebut yang akan disimpan pada PC (*Personal Computer*). Dimana *file* video yang dibutuhkan harus memiliki resolusi yang tepat dan memiliki kualitas yang bagus. Akan tetapi, permasalahannya adalah ukuran untuk resolusi *file* yang dibutuhkan memiliki ukuran yang cukup besar untuk ruang penyimpanan oleh karena itu diperlukan adanya suatu teknik untuk memampatkan file menjadi lebih kecil dan efesien dalam penyimpanannya serta mempersingkat waktu pertukarannya yang disebut dengan kompresi *file.*

Pada tahun 2019, Desvika Riyansyah pernah melakukan penelitian tentang kompresi *file* video dengan menggunakan algoritma *Interpolative Coding.* Berdasarkan hasil penelitian yang dilakukan, penerapan algoritma *Interpolative Coding* telah membuktikan bahwa ukuran yang besar dari suatu *file* video dapat dikompres menjadi *file* yang memiliki ukuran lebih kecil dari sebelumnya. Berdasarkan dari hasil pengujian terhadap sistem bahwa ukuran *file* video lebih kecil setelah dilakukan kompresi dengan rasio 47%[1].

Untuk mengatasi permasalahan dari besarnya ukuran *file* video yang didapat dengan format MP4 maka teknik kompresi diperlukan dengan tujuan memperkecil ukuran dari *file* video tersebut. Kompresi memiliki berbagai jenis algoritma yang dapat digunakan. Maka dari itu penulis bermaksud menggunakan algoritma *Prefix Codes* untuk mengompresi *file* video hasil ekstra aplikasi *kinemaster* berformat MP4 tersebut. Diharapkan dengan memperkecil ukuran dari *file* yang ada dapat mengefisiensikan ruang penyimpanan.

Algoritma *Prefix codes* adalah jenis sistem kode yang dibedakan berdasarkan kepemilikannya atas "properti awalan", yang mensyaratkan bahwa tidak ada keseluruhan kata kode dalam sistem yang merupakan awalan (segmen awal) dari setiap kata kode lain dalam sistem[2]. Tujuan dari algoritma ini untuk mengurangi ukuran *file* video yang telah diedit pada aplikasi *Kinemaster*, sehingga menghasilkan *file* yang berukuran lebih kecil dalam PC.

BIOSTech: Bulletin of Computer Science and Information Technology
Vol 1, No 1, Desember 2022
Hal: 22-29
Available Online at https://journal.grahamitra.id/index.php/biostech

2. METODOLOGI PENELITIAN

2.1 Kompresi

Kompresi merupakan proses yang mengarah pada minimisasi jumlah bit adalah teknik memperkecil atau memadatkan *file* yang berukuran besar menjadi lebih kecil dan mengurangi kebutuhan ruang penyimpanan. Proses untuk representasi digital seperti gambar, audio, dan video, yang menghasilkan ukuran data yang lebih kecil namun tetap menjaga kuantitas informasi dalam data tersebut[1]. Kompresi video adalah proses memperkecil atau meminimalisasi jumlah tiap bit yang merepresentasikan suatu video dengan data yang menjadi lebih kecil. Kompresi data juga diartikan sebagai teknik untuk memampatkan data agar diperoleh data dengan ukuran yang lebih kecil dari pada ukuran aslinya sehingga lebih efisien dalam menyimpannya atau mempersingkat waktu pertukaran data tersebut.

2.2 Algoritma *Prefix Codes*

Prefix Codes adalah jenis sistem kode yang dibedakan oleh kepemilikan "*prefixproperty*", yang mensyaratkan bahwa tidak ada kata kode keseluruhan dalam sistem yang merupakan awalan (awal segmen) dari kode kata lain dalam system. Terdapat empat kode dalam *prefix code* yaitu C1, C2, C3, C4 dan kode yang penulis gunakan adalah kode C1. Untuk mendapatkan nilai kode C1, C2, C3, dan C4 sebelumnya harus mengetahui nilai dari *Unary Code*, B(n) dan $\overline{B}(n)$. *Unary Code* dari bilangan positif n didefinisikan sebagai n – 1 diikuti oleh satu 0 atau alternatifnya seperti angka nol n – 1 diikuti dengan satu seperti tabel di bawah ini.

Tabel 1. *Unary Code*

N	Code	Alt. Code
1	0	1
2	10	01
3	110	001
4	1110	0001
5	11110	00001

Sumber : David Salomon, 2011[4].

B(n) untuk menunjukkan representasi biner dari bilangan bulat n, sementara $\overline{B}(n)$ untuk menunjukkan nilai B(n) tanpa bit dengan menghilangkan nilai bit 1 depan setiap bilangan B(n). Setelah menemukan nilai B(n) dan $\overline{B}(n)$ barulah mencari nilai kode C1 dengan contoh n = 5 = 101_2, ukuran B(5) adalah 3 lalu mulai dengan kode unari 110 (atau 001) dan menambahkan $\overline{B}(5)$ =01, dengan demikian kode lengkapnya adalah 110 | 01 (atau 001 | 10). Untuk mencari nilai kode C2, C3, dan C4 bisa dilihat pada tabel di bawah ini.

Tabel 2. Tabel Ketentuan *Prefix Codes*

N	Unary	B(n)	$\overline{B}(n)$	C1	C2	C3	C4
1	0	1		0|	0	0|	0
2	10	10	0	10|0	100	100|0	10|0
3	110	11	1	10|1	110	100|1	11|0
4	1110	100	00	110|00	10100	110|00	10|100|0
5	11110	101	01	110|01	10110	110|01	10|101|0
6	111110	110	10	110|10	11100	110|10	10|110|0
7	1111110	111	11	110|11	11110	110|11	10|111|0
8		1000	000	1110|000	1010100	10100|000	11|1000|0
9		1001	001	1110|001	1010110	10100|001	11|1001|0
10		1010	010	1110|010	1011100	10100|010	11|1010|0
11		1011	011	1110|011	1011110	10100|011	11|1011|0
12		1100	100	1110|100	1110100	10100|100	11|1100|0
13		1101	101	1110|101	1110110	10100|101	11|1101|0

Sumber : David Salomon, 2011[4].

3. HASIL DAN PEMBAHASAN

3.1 Analisa Masalah

Salah satu permasalahan yang hadir di era milenial ini adalah semua aktivitas dan kebutuhan yang dilakukan banyak orang sangat bergantung pada teknologi dan dunia digital. Pada era yang sangat *modern* ini ketertarikan masyarakat akan *file* berupa video sangat meningkat yang mengakibatkan kebutuhan akan ruang penyimpanan juga sangat meningkat. Oleh karena itu dibutuhkan suatu teknik untuk memperkecil ukuran suatu file untuk meminimalisir ruang penyimpanan dan mempercepat waktu *transfer* data.

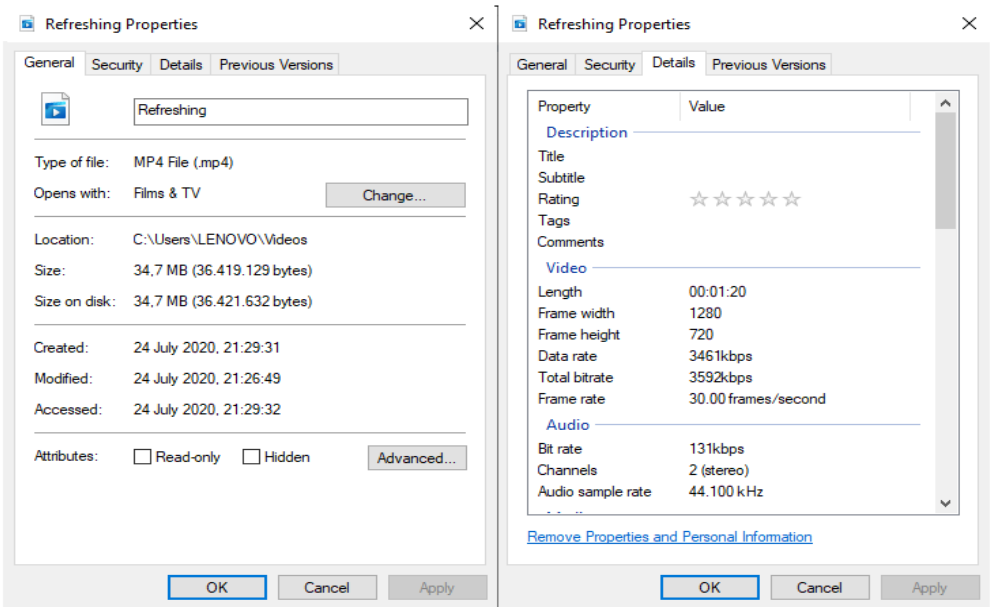
Analisa yang akan dilakukan pada penelitian ini adalah mengkompresi *file* video hasil ekstra aplikasi *kinemaster* berformat MP4 dengan menerapkan algoritma *prefix codes* dan menghasilkan file video terkompresi dengan ukuran yang lebih kecil. Aplikasi *kinemaster* adalah suatu aplikasi editor video profesional yang sangat lengkap yang dapat mengedit berbagai lapisan video gambar dan juga teks yang dilengkapi dengan pemotongan video yang tepat.

3.1.1 Penerapan Algoritma Prefix Codes

Prosedur atau langkah yang digunakan dalam algoritma *prefix codes* untuk mengkompresi *file* video yaitu, pertama membaca nilai dari bilangan heksadesimal dari *file* video tersebut dengan bantuan aplikasi *binary viewer*. Kemudian mengubah bilangan heksadesimal kedalam bilangan biner serta mengurutkan bilangan tersebut dari fekuensi tertinggi sampai yang ke frekuensi terendah. Setelah itu membentuk kode C1 algoritma *prefix codes*, aturan dalam pembentukan kode bilangan dengan menggunakan *prefix codes* dapat dilihat pada sub landasan teori bab sebelumnya. Langkah selanjutnya, dengan menggunakan kode C1 algoritma *prefix codes* bilangan biner dari *file* video tersebut di kompres mengikuti langkah-langkah yang ditentukan.

Pada penelitian ini, penulis akan membahas dua proses utama yaitu proses kompresi dan proses dekompresi, penulis akan mengkompresi sebuah video hasil ekstra dari aplikasi *kinemaster* dengan menggunakan algoritma *prefix codes. Prefix codes* merupakan salah satu algoritma yang bersifat *lossless* yang berarti pemampatan data tanpa menghilangkan data yang dimilikinya. Sebelum *file* dikompresi, terlebih dahulu dilakukan pembacaan biner yang terdapat pada *file* video untuk mendapatkan data berupa data biner. Untuk membaca nilai biner yang terdapat pada *file* video tersebut menggunakan aplikasi *Binary Viewer*. Dalam menganalisa *file* video harus dilakukan mengambilan *sample file* untuk mendapatkan nilai dari data pada sebuah *file* video yang berupa nilai *hexadecimal*. Berikut informasi objek *file* MP4 yang akan diambil *sample*nya sebelum dilakukan kompresi.

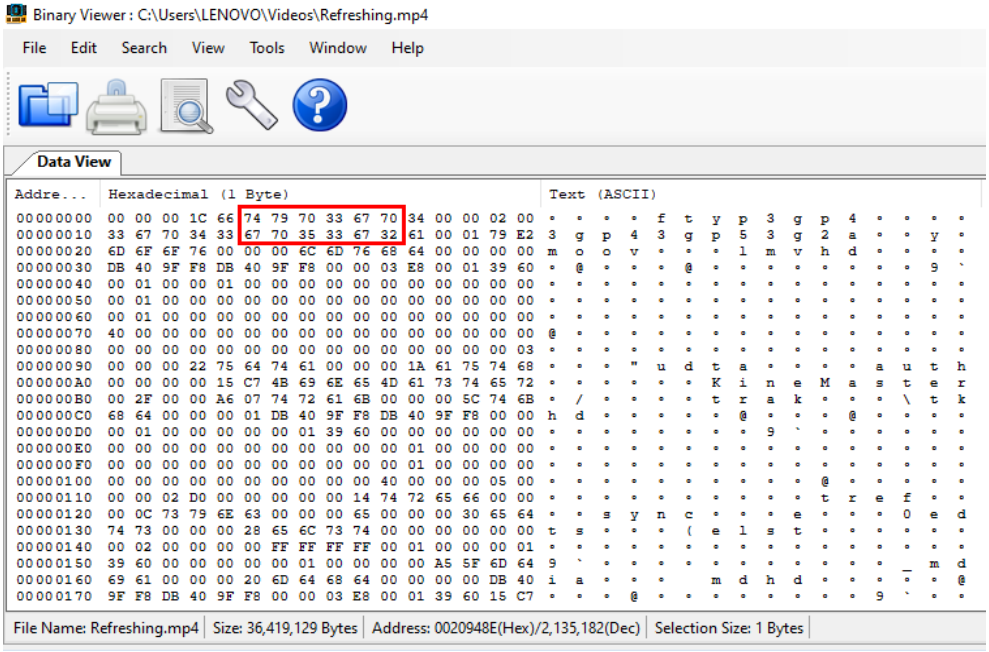
Gambar 1. File Video



Gambar 2. Informasi *Sampel File* MP4

Dari sampel f*ile* video didapat nilai *hexadecimal* menggunakan bantuan *software binary viewer* seperti pada gambar di bawah ini:

BIOSTech: Bulletin of Computer Science and Information Technology
Vol 1, No 1, Desember 2022
Hal: 22-29
Available Online at https://journal.grahamitra.id/index.php/biostech



Gambar 3. Kode Hexadesimal File Video Dari Hasil *Binary Viewer*

Berdasarkan gambar di atas didapat sampel nilai heksadesimal *file* Refreshing.mp4. Untuk keperluan hitungan manual hanya diambil sampel nilai sebanyak 12 karakter nilai heksadesimal *file* Refreshing.mp4. Adapun bilangan sampel heksadesimal tersebut adalah : 74 79 70 33 67 70 67 70 35 33 67 32.

Tabel 3. Tampilan Nilai Heksadesimal Berdasarkan *Biner Viewer*

74	79	70	33	67	70
67	70	35	33	67	32

Berikut langkah untuk mengkompresi dan mengkompresi *file* video :

1. Urutkan setiap simbol bilangan heksadesimal berdasarkan frekuensi kemunculan tertinggi.

Tabel 4. Pendataan Simbol

N	Heksadesimal	ASCII (*Binary*)	Frekuensi	Bit	Frekuensi x Bit
1	70	01110000	3	8	24
2	67	01100111	3	8	24
3	33	00110011	2	8	16
4	74	01110100	1	8	8
5	79	01111001	1	8	8
6	35	00110101	1	8	8
7	32	00110010	1	8	8
		Total Bit			96 bit

Berdasarkan tabel kode ASCII di atas, satu karakter bernilai delapan bit bilangan biner. Sehingga 16 bilangan desimal mempunyai nilai biner sebanyak 96 bit. Untuk mengubah satuan menjadi *byte* maka jumlah keseluruhan bit dibagikan 8, maka dihasilkan 96/8 = 12 *byte*.

2. Langkah selanjutnya adalah dengan membentuk tabel kode *C1* pada algoritma *prefix codes.* Adapun langkah dalam pembentukan *code word* C1 dengan menggunakan *prefix codes* adalah sebagai berikut:
   a. Untuk n=1, mengubah bilangan desimal 1 kedalam bentuk bilangan biner yaitu 1_2 maka ukuran dari B(1) = 1 jadi kita mulai dengan kode unari 0 dan membahkan nilai $\overline{B}(n)$. Untuk mendapatkan nilai $\overline{B}(n)$ adalah dengan cara menghilangkan angka 1 pada posisi depan dari biner,maka hasil dari $\overline{B}(1)$ =" " maka kode C1=0
   b. Untuk n=2, dengan nilai bilangan biner 10_2 maka ukuran dari B(2)=2 mulai dengan kode unari 10 dan membahkan nilai $\overline{B}(n)$= 0 maka kode C1=100
   c. Untuk n=3, dengan nilai bilangan biner 11_2 maka ukuran dari B(3)=2 mulai dengan kode unari 10 dan membahkan nilai $\overline{B}(n)$= 1 maka kode C1=101
   d. Untuk n=4, dengan nilai bilangan biner 100_2 maka ukuran dari B(4)=3 mulai dengan kode unari 110 dan membahkan nilai $\overline{B}(n)$= 00 maka kode C1=11000
   e. Untuk n=5, dengan nilai bilangan biner 101_2 maka ukuran dari B(5)=3 mulai dengan kode unari 110 dan membahkan nilai $\overline{B}(n)$= 01 maka kode C1=11001

f. Untuk n=6, dengan nilai bilangan biner 110_2 maka ukuran dari B(6)=3 mulai dengan kode unari 110 dan membahkan nilai $\overline{B}(n)$= 10 maka kode C1=11010
g. Untuk n=7, dengan nilai bilangan biner 111_2 maka ukuran dari B(5)=3 mulai dengan kode unari 110 dan membahkan nilai $\overline{B}(n)$= 11 maka kode C1=11011, maka kode *C1* dapat dilihat pada tabel 5. di bawah ini :

Tabel 5. Kode C1 Pada Algoritma *Prefix Codes*

N	Unary	B(n)	$\overline{B}(n)$	C1
1	0	1		0|
2	10	10	0	10|0
3	110	11	1	10|1
4	1110	100	0	110|00
5	11110	101	1	110|01
6	111110	110	10	110|10
7	1111110	111	11	110|11

Pada tabel kode *C1* pada algoritma *prefix codes* di atas hanya menunjukan kode sampai nilai karakter ke 7. Bagaimana proses yang dilakukan untuk mendapat kode *C1*, jika jumlah karakter yang dihasilkan lebih dari 7 karakter. Misalnya pada n = 15 dengan nilai biner dari 15 = 1111_2. Maka ukuran dari B(15) adalah 4, jadi kita mulai dengan kode unari 1110(atau 0001) dan menambahkan nilai dari $\overline{B}(n)$. Maka untuk mendapatkan nilai $\overline{B}(n)$ adalah dengan cara menghilangkan angka 1 pada posisi depan dari biner,maka hasil dari $\overline{B}(15)$=111, dengan demikian kode lengkap yang didapat adalah 1110 | 111 (atau 0001 | 111). Proses selanjutnya adalah melakukan kompresi nilai dari *hexadesimal* sampel dengan nilai kode *C1* yang didapat dari tabel 5. di atas. Adapun proses kompresi video Refreshing.mp4 dapat dilihat pada tabel di bawah ini :

Tabel 6. Hexadesimal Kode C1 Algoritma *Prefix Codes* Terkompresi

N	Nilai Hexadesimal	Frek	Codeword (C1)	Bit	Frek x Bit
1	70	3	0	1	3
2	67	3	100	3	9
3	33	2	101	3	6
4	74	1	11000	5	5
5	79	1	11001	5	5
6	35	1	11010	5	5
7	32	1	11011	5	5
		Total Bit			38 Bit

Berdasarkan tabel 6. dapat dilihat bahwa bentuk *string* bit dari nilai *hexadesimal* "74 79 70 33 67 70 67 70 35 33 67 32" setelah dikompresi yaitu "110001100101011000100011010101100111011" dengan jumlah 38 *bit*. Untuk diubah menjadi satuan *byte* maka dibagi 8 yaitu 38/8=4,75 *byte*. Kemudian melakukan pemeriksaan terhadap panjang *string bit* untuk menambahkan *padding* dan *flagging*. Jika sisa bagi panjang *string bit* terhadap 8 adalah 0 maka tambahkan 00000001 dan nyatakan dengan *bit* akhir. Jika sisa bagi panjang *string bit* terhadap 8 adalah *n* (1, 2, 3, 4, 5, 6, 7) maka tambahkan 0 sebanyak 7 – *n* + "1" di akhir *string bit*, nyatakan dengan L. Lalu tambahkan bilangan *biner* dari 9 – *n* nyatakan dengan *bit* akhir. Total jumlah bit yang telah dikompresi adalah 38 *bit*, maka *N* = *MOD* (38,8) = 6, dimana L = 7 – 6 + "1" = 1, maka tambahkan **01**. *Bit* akhir = 9 – 6 = 3, nilai binernya menjadi **00000011**. Sehingga *string bit* yang terbentuk adalah : "11000110010101100010001101010110011011**0100000011**".

Total panjang *bit* keseluruhan setelah ada penambahan *bit* adalah 38+2+8=48. Selanjutnya total keseluruhan *string bit* akan dibagi menjadi menjadi 8 *bit* dalam satu kelompok *string bit*, dan *string bit* yang sudah dikelompokkan akan diubah kedalam bilangan desimal, bilangan desimal yang diperoleh dari bilangan kelompok biner selanjutnya akan diubah menjadi berbagai karakter unik yang terdapat pada tabel ASCII.

Tabel 7. Pengelompokan Bit dan Pencarian Karakter Unik

No	*String Bit* Hasil Kompresi	Bilangan Hexadesimal	Karakter Unik
1	11000110	C6	Æ
2	01010110	56	V
3	00100011	23	#
4	01010110	56	V
5	01101101	6D	M
6	00000011	3	␣

Berdasarkan hasil kompresi *string* bit nilai *hexadesimal* dari sampel *file* video menggunakan algoritma *prefix codes* di atas, maka didapatkan hasil kompresi *string bit* yang lebih sedikit yaitu sebanyak 48 *string bit* dari *string bit* awal sebanyak 96 *string bit*, sebagai berikut : 11000110010101100010001101010110011011

BIOSTech: Bulletin of Computer Science and Information Technology
Vol 1, No 1, Desember 2022
Hal: 22-29
Available Online at https://journal.grahamitra.id/index.php/biostech

Untuk mengetahui tingkat kinerja suatu algoritma kompresi, maka kita harus mengukur hasil kompresi *file* video tersebut. Semakin kecil ukuran *file* yang dikompresi berarti semakin berkualitas tingkat kinerja kompresinya begitu pula sebaliknya. Dalam menganalisis algoritma kompresi parameter yang akan dipakai penulis sebagai tolak ukur untuk menentukan kinerja algoritma *Prefix Codes* adalah *Ratio of Compression, Compression Ratio* dan *Space Saving.*

1. *Ratio of Compression* (R_C)
   Ratio of Compression (R_C) adalah nilai perbandingan antara ukuran bit data sebelum dikompresi dengan ukuran bit data setelah dikompresi.
   $$R_C = \frac{Jumlah\ bit\ sebelum\ dikompresi}{Jumlah\ bit\ setelah\ dikompresi}$$
   $$R_C = \frac{96\ bit}{48\ bit}$$
   $$R_C = 2$$
2. *Compression Ratio* (C_R)
   Salah satu parameter pengukur kinerja algoritma adalah hasil perbandingan antara ukuran data sebelum dikompresi dengan ukuran data setelah dikompresi pada data yang terkompresi yang biasa disebut *Compression Ratio* dengan cara kerja sebagai berikut :
   Ukuran data sebelum dikompresi = 96/8=12 *byte*
   Ukuran data sesudah dikompresi = 48/8=6 *byte*
   $$C_R = \frac{ukuran\ data\ sesudah\ dikompresi}{ukuran\ data\ sebelum\ dikompresi} \times 100\%$$
   $$C_R = \frac{6\ byte}{12\ byte} \times 100\%$$
   $$C_R = 50\%$$
3. *Space Savings* (SS)
   Space Saving (SS) selisih antara data yang belum dikompresi dengan besar data yang sudah dikompresi
   SS = 100% - C_R
   SS = 100% - 50% = 50%

Berdasarkan hasil kompresi *string* bit sampel nilai *hexadesimal* dari *file* Refrshing.MP4 menggunakan algoritma *prefix codes* di atas, maka didapatkan hasil kompresi *string bit* yang lebih sedikit yaitu sebanyak 48 *bit* (6 *byte*) dari *string bit* awal sebanyak 96 *bit* (12 *byte*) *bit.* Dari hasil perhitungan tersebut diperoleh hasil *Ratio of Compression* (R_C) sebanyak 2 *bit, Compression Ratio* (C_R) 50% dan *Space Savings* (SS) 50%. Dengan kata lain *string* bit nilai *hexadesimal* sampel *file* video hasil ekstra aplikasi *kinemaster* setelah dikompresi memiliki ukuran yang lebih kecil dari ukuran sebelum di kompresi.

Proses dekompresi yang dilakukan adalah menganalisa semua *bit* hasil dari kompresi sebelumnya. Adapun *bit* keseluruhan hasil kompresi dapat dilihat pada tabel berikut :

Tabel 8. Nilai Hexadesimal Video Terkompresi

No	*String Bit* Hasil Kompresi	Bilangan Heksadesimal	Karakter Unik
1	11000110	C6	Æ
2	01010110	56	V
3	00100011	23	#
4	01010110	56	V
5	01101101	6D	M
6	00000011	3	␣

Berdasarkan tabel di atas maka diambil seluruh nilai biner dan digabungkan menjadi 11000110010101100010001101010110011011**010000001.** Selanjutnya adalah dengan mengembalikan *binary* menjadi *string bit* semula dengan menghilangkan biner *padding* dan *flagging.* Untuk mengembalikan *binary* menjadi *string bit* semula dapat dilakukan melalui langkah berikut ini. Lakukan pembacaan pada 8 *bit* terakhir, hasil pembacaan berupa bilangan heksadesimal. Nyatakan hasil pembacaan dengan *n*, hilangkan *bit* pada bagian akhir sebanyak 7+*n*, maka dari hasil pembacaan 8 *string bit* terakhir diperoleh *n*=3 sehingga 7+3=10. Selanjutnya mengurangi *string bit* terakhir sebanyak 10 *bit*, hasil pengembalian *binery* menjadi *string bit* semula dapat dilihat sebagai berikut ini : 11000110010101100010001101010110011011. Selanjutnya membaca ataupun memerikasa *string bit* dari awal sampai ditemukan karakter kode yang sesuai dengan tabel Kode *C1* Pada algoritma *Prefix Codes.* Berikut adalah tabel pencocokan nilai *bit* :

Tabel 9. Pencocokan Nilai *Bit*

Indeks	Pencocokan *bit*	Keterangan	Nilai Heksadesimal
1.	1	Tidak ada	
2.	11	Tidak ada	
3.	110	Tidak ada	

Indeks	Pencocokan *bit*	Keterangan	Nilai Heksadesimal
4.	1100	Tidak ada	
5.	11000	Ada	74
6.	1	Tidak ada	
7.	11	Tidak ada	
8.	110	Tidak ada	
9.	1100	Tidak ada	
10.	11001	Ada	79
11.	0	Ada	70
12.	1	Tidak ada	
13.	10	Tidak ada	
14.	101	Ada	33
15.	1	Tidak ada	
16.	10	Tidak Ada	
17.	100	Ada	67
18.	0	Ada	70
19.	1	Tidak Ada	
20.	10	Tidak Ada	
21.	100	Ada	67
22.	0	Ada	70
23.	1	Tidak Ada	
24.	11	Tidak Ada	
25.	110	Tidak Ada	
26.	1101	Tidak Ada	
27.	11010	Ada	35
28.	1	Tidak Ada	1
29.	10	Tidak Ada	
30.	101	Ada	33
31.	1	Tidak Ada	
32.	10	Tidak Ada	
33.	100	Ada	67
34.	1	Tidak Ada	
35.	11	Tidak Ada	
36.	110	Tidak Ada	
37.	1101	Tidak Ada	
38.	11011	Ada	32

Maka berdasarkan hasil dekompresi *Prefix codes* dengan kode C1 di atas didapati nilai heksadesimal awal sebelum kompresi sebagai berikut :

Tabel 10. Hasil Dekompresi

No	Nilai Heksadesimal	Codeword (C1)
1	70	0
2	67	100
3	33	101
4	74	11000
5	79	11001
6	35	11010
7	32	11011

4. KESIMPULAN

Berdasarkan dari penelitian yang telah dilakukan, maka hasil akhir dari penelitian tersebut dapat diambil beberapa kesimpulan. Adapun kesimpulan tersebut Berdasarkan prosedur kompresi dengan menggunakan algoritma *Prefix Codes* telah berhasil melakukan proses kompresi *file* video hasil ekstra aplikasi kinemaster berekstensi *.MP4 sehingga proses kompresi dapat berjalan sesuai dengan teknik kompresi. Berdasarkan penerapan algoritma *Prefix Codes* telah membuktikan bahwa suatu *file* video yang memiliki ukuran besar dapat dikompres menjadi ukuran yang lebih kecil. Berdasarkan dari hasil pengujian terhadap sistem bahwa ukuran *file* video lebih kecil setelah dilakukan kompresi.

REFERENCES

[1] D. Riyansyah, "Perancangan Aplikasi Kompresi File Video Menggunakan Algoritma Interpolative Coding," *KOMIK (Konferensi Nas. Teknol. Inf. dan Komputer)*, vol. 3, no. 1, pp. 392–397, 2019, doi: 10.30865/komik.v3i1.1618.
[2] D. Salomon, *Handbook of Data Compression*, vol. 53, no. 9. 2008.

[3] [F. Gram et Al., "Analisis Perbandingan Kompresi dan Dekompresi Menggunakan Algoritma Shannon," vol. 3, 2016.
[4] C. S. Kim and C. C. J. Kuo, *Data Compression*, vol. 1. 2011.
[5] K. U. and B. S. Negara, "Deteksi Obyek Manusia Pada Basis Data Video Menggunakan Metode Background Subtraction Dan Operasi Morfologi," vol. 2, 2016.
[6] Y. S. et Al, "Perancangan Aplikasi Edukatif Berbasis Multimedia," vol. 1, 2014.
[7] S. Panggang and D. I. Yogyakarta, "PEMBELAJARAN TEKS REPORT DENGAN PROYEK ' CERDIG ' BERBASIS KINEMASTER Laily Amin Fajariyah REPORT TEXT WRITING THROUGH CERDIG PROJECT WITH KINEMASTER Laily Amin Fajariyah SMPN 5 Panggang , D . I . Yogyakarta."
[8] A. Nugroho, "Rekayasa Perangkat Lunak Berorientasi Objek dengan Metode USDP(Unfied Software Development Process," 2010.
[9] M. S. Rosa A.S, "Rekayasa Perangkat Lunak Terstruktur dan Berorientasi Objek," Bandung, 2014.
[10] R. Priyanto, *Langsung Bisa Visual Basic.NEt 2008*. Yogyakarta, 2009.
[11] Ihsan and D. P. Utomo, "Analisis Perbandingan Algoritma Even-Rodeh Code Dan Algoritma Subexponential Code Untuk Kompresi File Teks," KOMIK (Konferensi Nas. Teknol. Inf. dan Komputer), vol. 4, no. 1, 2020.
[12] S. R. Saragih and D. P. Utomo, "Penarapan Algoritma Prefix Code Dalam Kompresi Data Teks," KOMIK (Konferensi Nas. Teknol. Inf. dan Komputer), vol. 4, no. 1, 2020.
[13] 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, 2020.