BIMASATI (Bulletin of Multi-Disciplinary Science and Applied Technology)
ISSN : 2809-6096 (Media Online)
Vol 1, No 2, Februari 2022
Hal : 41-48

Perancangan Aplikasi Kompresi File Video Dengan Menggunakan Algoritma *Start Step Stop Code*

Joni Alpensius Simatupang, Guidio L. Ginting, Fince Tinus Waruwu

Fakultas Ilmu Komputer dan Teknologi Informasi, Program Studi Teknik Informatika, Universitas Budi Darma, Medan, Indonesia
Email: jonialpensius24@gmail.com

Abstrak−File video memiliki ukuran yang besar sehingga mempengaruhi ruang penyimpanan dan proses pengiriman. ukuran inilah yang menjadi faktor berapa banyak ruang penyimpanan yang akan dipakai dan berapa lama waktu yang dibutuhkan untuk melakukan pengiriman file. Semakin panjang file video maka ukuran juga semakin besar ukuran yang besar dapat diatasi dengan cara melakukan proses kompresi. Kompresi dilakukan untuk mengurangi ukuran dari sebuah file. Algoritma kompresi yang dapat digunakan yaitu algoritma start step stop code. Kelebihan algoritma start step stop code yang pertama menentukan nilai n = 0, kemudian tentukan nilai a = start + n x stop, selanjutnya subset yang dimulai dengan n akan diawali oleh angka '1' dan diikuti oleh sisipan '0' lalu diikuti oleh kombinasi dari bit a.

Kata Kunci: Start Step Stop Code; Kompresi;Video

Abstract−Video files have a large size that affects the storage space and the delivery process. This size is a factor in how much storage space will be used and how long it will take to send files. The longer the video file, the larger the size can be overcome by doing the compression process. Compression is done to reduce the size of a file. The compression algorithm that can be used is the start step stop code algorithm. The advantage of the start step stop code algorithm is that it first determines the value of n = 0, then determines the value of a = start + nx stop, then the subset starting with n will be preceded by the number '1' and followed by the insertion of '0' followed by a combination of bits. a.

Keywords: Start Step Stop Code; Compression; Video

1. PENDAHULUAN

Saat ini perkembangan dunia teknologi semakin menyebar luas dengan kebutuhan masyarakat dalam memperoleh informasi secara cepat. Berbagai macam fasilitas teknologi terus dikembangkan agar masyarakat dapat melakukan pertukaran informasi dalam bentuk teks, gambar, audio dan video dengan baik. Semakin tingginya permintaan informasi yang *real-time,* menjadikan perpindahan data harus semakin cepat dan akurat. Hal tersebut menjadi masalah, dimana jalur komunikasi di Indonesia khususnya internet masih berada dalam kategori lambat dan sering bermasalah, dengan demikian data yang berukuran kecil akan dipilih karena akan lebih cepat dikirim dan lebih menghemat tempat penyimpanan.

Perpindahan data sangat mudah dilakukan pada saat ini, tetapi tempat penyimpanannya yang menjadi kendala sangat mendasar. Video sebenarnya berasal dari bahasa Latin, video-vidisium yang artinya melihat atau mempunyai daya penglihatan. Video juga merupakan salah satu jenis media audio visual. Hal inilah yang menyebabkan *file* video harus dimampatkan agar ukurannya menjadi lebih kecil. Teknik pemampatan data ini disebut dengan teknik kompresi data. Adapun tujuan dari kompresi data adalah untuk mengurangi ukuran *file* video sebelum menyimpan atau memindahkan data ke dalam media penyimpanan. Video pada umumnya berisi rangkaian karakter dan dapat membentuk suatu kata, surat dan narasi. Misalnya, *file* video yang berekstensi (*.doc) yang ukurannya besar mengakibatkan proses pengiriman semakin lama, serta menggunakan ruang memori yang besar dalam penyimpanan nya. Maka untuk menghemat ruang penyimpanan dan mempercepat proses pengiriman *file* video, perlu dilakukan proses kompresi agar ukuran *file* video tersebut menjadi lebih kecil.

Kompresi Data adalah proses mengubah sebuah aliran data input menjadi aliran data baru yang memiliki ukuran lebih kecil. Aliran yang dimaksud adalah berupa *file* ataupun *buffer* dalam memori. Terdapat banyak metode untuk kompresi data. *Lossy* dan *Lossless Compression* adalah pengelompokan metode kompresi berdasarkan keutuhan data.

Lossy Compression menghilangkan beberapa data untuk memperoleh kompresi yang lebih baik, seperti *Linear Predictive Coding*, *A-LawAlgorithm*, *Mu-LawAlgorithm*, *Fractal Compression*, dan lain-lain. Ketika proses dekompresi dilakukan, maka output yang dihasilkan tidak sama dengan data aslinya. Metodekompresi ini lebih efektif untuk mengkompresi data gambar, video, atau audio. Sedangkan *LosslessCompression* tidak menghilangkan data sama sekali, sehingga cocok untuk mengkompresi data berupa teks. Yang termasuk *LosslessCompression* adalah *Burrows-Wheeler, DEFLATE, LZW, FLBE, VLBE, Huffman, PPM, Shanon-Fano,Sequitur,* dan lain-lain. Dekompresi Data merupakan Suatu Data yang sudah dikompresi tentunya harus dapat dikembalikan lagi kebentuk aslinya, prinsip ini dinamakan dekompresi[1].

Berdasarkan penelitian terdahulu yang dilakukan oleh Khairunnada kompresi data dengan judul "Analisis Perbandingan Kinerja Algoritma *Start-Step-Stop Code* dan *Gopala-Hemachandra Code 2 (GH-2(n))* pada Kompresi *File* Teks" menyimpulkan bahwa dalam melakukan kompresi *file* teks terhadap string homogen, algoritma *Start-Step-Stop Code* dan algoritma *Gopala-Hemachandra Code 2 (GH-2(n))* menghasilkan nilai yang sama pada *Compression Ratio, Space Savings*, dan *Bit Rate* untuk setiap jumlah stringnya karena kode pertama pada algoritma *Start-Step-Stop Code* dan algoritma *Gopala-Hemachandra Code 2 (GH-2(n))* sama-sama berjumlah 4 bit. Cara kerja algoritma *Start-Step-Stop Code* adalah, yang pertama menentukan nilai $n = 0$, kemudian tentukan nilai $a = start + n$ x *stop,* selanjutnya *subset* yang dimulai dengan *n* akan diawali oleh angka '1' dan diikuti oleh sisipan '0' lalu diikuti oleh kombinasi dari bit *a*[2]. Pada

penelitian berikutnya yang dilakukan oleh Jeffrey N. Denenberg, Edward D. Weinberger, Michael L. Gordon, kompresi data dengan judul "Data Compression Method for use in a Computerized Informational and Transactional Network" menyimpulkan bahwa pada penelitian tersebut menemukan beberapa keuntungan setelah mereka menggunakan metode *Star-Step-Stop Code*, berikut adalah beberapa kesimpulan dan keuntungan dari penelitiannya: mengurangi transmisi data dan persyaratan penyimpanan yang interaktif jaringan layanan tanpa secara substansial, untuk digunakan dalam antar jaringan layanan aktif yang tidak mahal untuk dipasang dan beroperasi, mudah diperbarui untuk mengakomodasi perubahan dalam data yang dikirimkan dan disimpan dalam layanan interaktif, untuk menentukan frekuensi terjadinya byte-pair frekuensi dalam aliran data jaringan[3].

Dalam penelitian ini proses yang akan dilakukan adalah dengan mengkompresi *file* video dengan algoritma *Start-Step-Stop-Code* dengan menghasilkan sebuah *file* yang sudah terkompresi, kemudian dalam mengembalikan *file* seperti semula akan memulai proses dekompresi.

2. METODOLOGI PENELITIAN

2.1 Kompresi

Kompresi adalah proses mengubah sebuah aliran data *input* menjadi aliran data baru yang memiliki ukuran lebih kecil. Aliran yang dimaksud adalah berupa *file* ataupun *buffer* dalam memori. Terdapat banyak metode untuk kompresi data. *Lossy compression* dan *Lossless compression* adalah pengelompokan metode berdasarkan ketentuan data.

Lossy Compression menghilangkan beberapa data untuk memperoleh kompresi yang lebih baik, seperti *Linier Predictive Coding, A-Law Algorithm, Mu-Law Algorithm, Fractal Compression* dan lain-lain. Ketika proses kompresi dilakukan maka *output* yang dihasilkan tidak sama dengan data aslinya. Metode kompresi ini lebih efektif untuk mengkompresi data gambar, Video, atau audio. Sedangkan *Lossless Compression* tidak menghilangkan data sama sekali, sehingga cocok untuk data berupa teks. Yang termasuk *Lossless Compression* adalah *Burrows-Wheeler, DEFLATE, LZW, FLBE, VLBE, Huffman, PPM, Shanno-Fano, Sequitur* dan lain-lain. Dekompresi data merupakan suatu data yang dikompresi tentunya harus dapat dikembalikan lagi kebentuk aslinya, prinsip ini dinamakan dekompresi [3].

2.2 Algoritma *Start Step Stop Code*

Algoritma *Start Step Stop Code* didiusulkan oleh Fialan dan Greene, dimana algoritma ini bergantung pada nilai *Start Step Stop Code*, dan *Stop* sebagai t*riplet* parameter untuk bilangan bulat *n* non negative (Salomo, 2007). Langkah-langkah untuk membangun *set of bits* dari *Start Step Stop Code* adalah sebagai berikut:

1. Tentukan nilai *n = 0.*
2. Tentukan nilai a = *start* + *n* x *step*
3. Setiap *subset* yang dimulai dengan n akan diawali oleh angka "1" dan diikuti oleh sisipan "0", lalu diikuti oleh kombinasi dari bit a. sehingga didapat nilai "2"
4. Lakukan penambahan nilai n pada langkah selanjutnya.
   a. Jika n < *stop*, maka pergi ke langkah ke 2.
   b. Jika n > *stop*, maka akan muncul pesan error dan program akan berhenti
   c. Jika n = *stop*, maka ulangi langkah 2 dan 3 tanpa ada sisipan 0 bit dari langkah 3.

Langkah – langkah ini membuat ketiga parameter harus memilih berdasarkan *start* + n x *step* dan mencapai "*stop*" untuk bilangan bulat *n* non negative (salomon, 2007). Berdasarkan langkah-langkah tersebut, pada table 2.1 disajikan range untuk kode *Start Step Stop Code* dengan parameter start = 3 step =2 dan stop = 9[6].

Table 1. *Range Start-Step-Stop Code* (3,2,9)

N	***a* = 3+ *n*.2**	***Nth codeword***	***Number of codeword***	***Range of integers***
0	3	0xxxxx	23 = 8	0 - 7
1	5	0xxxxxxx	25 = 32	8 - 39
2	7	0xxxxxxxxx	27 = 128	40 – 167
3	9		29 = 512	168 - 679
		Total	680	

3. HASIL DAN PEMBAHASAN

3.1 Analisa Masalah

Analisa sistem merupakan tahap awal dalam sebuah penelitian yang bertujuan mengetahui masalah terkait dalam pembuatan sebuah sistem dan menggambarkan proses yang ada didalam sistem untuk mengetahui keluaran yang sesuai dengan kebutuhan pemakai (*user*). Analisan terhadap system merupakan tahap yang sangat penting dalam aplikasi yang akan dirancang, dalam hal ini peniliti menggunaka algoritma *Start Step Stop Code* untuk mengkompresi *file* video. Pengompresan *file* berguna untuk mengurangi media penyimpanan serta mempercepat prosess pengiriman suatu *file* video.

File video yang akan dianalisa yaitu *file* video berformat MP4 mempunyai ukuran yang cukup besar. Format MP4 merupakan format minim kompresi, sehingga ukurannya cukup besar dan memerlukan pengkompresian *file*. Adapun tujuan dari analisa terhadap sistem yang dirancang yaitu untuk mengetahui kebutuhan dari sistem serta membantu meminimalisir sumber daya yang berlebihan.

Dalam proses kompresi dilakukan penerapan algoritma *Start Step Stop Code* terhadap *file* video yang memiliki ukuran yang relatif besar. Kompresi data merupakan teknik pemampatan data, sehingga diperoleh ukuran *file* yang lebih kecil dari aslinya. *File* video memiliki ukuran relatif besar, semakin lama durasi *file* video maka semakin besar alokasi penyimpanan yang dibutuhkan.

3.1.1. Penerapan Algoritma Start Step Stop Code

Dalam penelitian ini, penulis akan membahas 2 proses utama yaitu proses kompresi dan proses dekompresi, penulis akan mengkompresi sebuah *file* video dengan menggunakan algoritma *Start Step Stop Code* merupakan salah satu algoritma yang termasuk didalam kompresi *lossy*. File video yang akan dikompresi adalah *file* mp4 yang berdurasi 55 detik dengan besar ukuran *file* 5.88 MB, maka sebelum *file* dikompresi, terlebih dahulu dilakukan pembacaan bilangan *hexa* yang terdapat pada *file* video untuk mendapatkan data berupa data biner. Membaca biner yang terdapat pada *file* video menggunakan aplikasi *Binary Viewer* untuk mancari nilai biner pada *file* video. Berikut adalah contoh *file* video yang akan dikompres dan didekompresi.

Tabel 1. Tampilan Nilai *Hexa* Berdasarkan *Biner Viewer*.

00	**00**	**00**	**20**	**66**	**74**	**79**	**70**
69	**73**	**6F**	**6D**	**69**	**73**	**6F**	**32**

Berdasarkan pada tabel diatas maka diperoleh nilai *Hexadesimal* sebagai berikut 00,00,00,20,66,74,79,70,69,73,6F,6D,69,73,6F,32, kemudian nilai *hexa* dimasukan kedalam table untuk melakukan pembacaan frekuensi. Pembacaan frekuensi dilakukan dengan menghitung jumlah nilai yang sama disetiap nilai *hexa* yang muncul. Adapun pembacaan frekuensi dapat dilihat pada tabel dibawah ini :

Tabel 2. Pembacaan Nilai Heksa

No	Heksa	Frekuensi
1	00	3
2	20	1
3	66	1
4	74	1
5	79	1
6	70	1
7	69	2
8	73	2
9	6F	2
10	6D	1
11	32	1
	Jumlah	16

Berdasarkan pada tabel diatas, didapatkan beberapa nilai *heksa* yang sama. Sebelum proses kompresi, langkah awal adalah membaca nilai heksa kemudian membuat table nilai *hexa* yang diurutkan dari nilai frekuensi terbesar (nilai Heksa yang sama) ke terkecil. Urutan nilai *heksa* dapat dilihat pada tabel dibawah ini :

Tabel 3. *Heksa* Yang Belum Dikompresi

No	Nilai Heksa	Binary	Bit	Frek	Bit x fre
1	00	00000000	8	3	24
2	20	01001001	8	1	8
3	66	01000100	8	1	8
4	74	00110011	8	1	8
5	79	00000100	8	1	8
6	70	00010010	8	1	8
7	69	00000011	8	2	16
8	73	01101101	8	2	16
9	6F	01100001	8	2	16
10	6D	01101010	8	1	8
11	32	00100000	8	1	8
		Total Bit			128 Bit

ISSN : 2809-6096 (Media Online)
Vol 1, No 2, Februari 2022
Hal : 41-48

Berdasarkan table diatas, Langkah – langkah ini membuat ketiga parameter harus memilih berdasarkan *start* + n x *step* dan mencapai "*stop*" untuk bilangan bulat *n* non negative. Berdasarkan langkah-langkah tersebut, disajikan range untuk kode *Start Step Stop Code* dengan parameter start = 3 step =2 dan stop = 9

Table 4. *Range Start-Step-Stop Code*

N	***a* = 3+ *n*.2**	***Nth codeword***	***Number of Codeword***	***Range of Integers***
0	3	0xxxxx	23 = 8	0 – 7
1	5	0xxxxxxx	25 = 32	8 – 39
2	7	0xxxxxxxxx	27 = 128	40 – 167
3	9		29 = 512	168 – 679
		Total	680	

Hasil dari algoritma *Start Step Stop Code* pada nilai heksa 00, 20, 66, 74, 79, 70, 69, 73, 6F, 6D, 32 dapat dilihat pada table dibawah :

Tabel 5. yang belum dikompres

Heksa	**Frekuensi**	**ASCII Desimal**	**ASCII Binary**	**Bit**	**Bit x Frek**
00	3	0	00000000	8	24
20	2	32	00100000	8	8
66	2	102	01100110	8	8
74	2	116	01110100	8	8
79	1	121	01111001	8	8
70	1	112	01110000	8	8
69	1	105	01101001	8	16
73	1	115	01110011	8	16
6F	1	111	01101111	8	16
6D	1	109	01101101	8	8
32	1	50	00110010	8	8
		Jumlah Bit x Frekuensi			128 Bit

Berdasarkan tabel diatas *file* yang sudah dikompresi, dibawah ini merupakan Heksa yang sudah dikompresi dengan menerapkan rumus dari pada algoritma *start step stop* tersebut.

Table 6. Heksa yang sudah dikompresi dengan algoritma

Heksa	**Frekuensi**	***Start Step Stop Code* (3,2,9)**	**Bit**	**Frek x Bit**
00	3	0000	4	12
20	2	0001	4	8
66	2	0010	4	8
74	2	0011	4	8
79	1	0100	4	4
70	1	0101	4	4
69	1	0110	4	4
73	1	0111	4	4
6F	1	10000000	8	8
6D	1	10000001	8	8
32	1	00010001	9	9
		Jumlah Frekuensi x Bit		77

Setelah kode berhasil digenerate berdasarkan perhitungan algoritma *Start Step Stop Code*, tahap selanjutnya adalah menyusun kembali kode -kode yang telah dihasilkan dari proses kompresi sesuai dengan posisi karakter pada nilai heksa **"00, 00, 00, 20, 66, 74, 79, 70, 69, 73, 6F, 6D, 69,73, 6F, 32"** seperti yang disajikan. Berdasarkan Data pada table string bit hasil kompresi dengan algoritma *Start Step Stop Code* dapat dituliskan sebagai berikut :

00	**00**	**00**	**20**	**66**	**74**
0000	0001	0010	0001	0010	0011
79	**70**	**69**	**73**	**6F**	**6D**
0100	0101	0110	0111	10000000	10000001
69	**73**	**6F**	**32**		
0110	0111	10000000	00010001		

Berdasarkan table *String Bit* hasil kompresi dengan algoritma *Start Step Stop Code* dapat dituliskan sebagai berikut :

0000	0001	0010	0001	0010	0011	0100
0101	0110	0111	10000000	10000001	0110	0111
10000000	10000001					

Kemudian sebelum dapatkan hasil keseluruhan akhir kompresi dilakukan penambahan *String* bit itu sendiri yaitu *padding* bit dan *flaging* bit. Jika sisa bagi panjang string bit terhadap 8 adalah 0 maka 000000001. Nyatakan dengan bit akhir. Sedangkan jika sisa bagi panjang string bit terhadapat 8 adalah n (1,2,3,4,5,6,7) maka tambahkan sebanyak 7 – *n* = "1" di akhir string bit. Nyatakan L. lalu tambahkan bilangan biner dari 9 – n. nyatakan dengan bit akhir. Karna jumlah string 77 habis dibagi 8 maka kita hanya menambahkan bilangan biner dari 9 – n. nyatakan dengan bit akhir.

7-n + "1"
7-5 + "1" = 001
Bit Akhir 9 – n
Bit Akhir = 9 – 5 = 4= **00000100**

Gambar 1. perhitungan Penambahan Bit

0000 0001 0010 0001 0010 0011 0100
0101 0110 0111 1000000 10000001 0110
0111 10000000 10000001 00000100

Gambar 2. *String* Bit Yang telah ditambahkan

Total panjang *bit* keseluruhan setelah ada penambahan *bit* adalah 77 + 3 + 8 = 88. Selanjutnya lakukan pemisahan *bit* menjadi beberapa kelompok. Setiap kelompok terdiri dari 8 *bit* seperti pada gambar dibawah ini :

00000001	00100001	00100011	01000101	01100111
10000000	10000001	01100111	10000000	10000001
00000100				

Gambar 3. pembagian *String bit*

Berdasarkan pada pembagian kelompok nilai biner, didapatkan 10 kelompok nilai biner baru yang sudah terkompresi beserta nilai biner penambahan bit. Setelah pembagian dilakukan, maka pixel yang sudah dibagi dirubah kedalam suatu karakter dengan terlebih dahulu mencari nilai decimal dari string bit tersebut menggunakan kode ASCII untuk mengetahui nilai *pixel* yang sudah terkompresi. Adapun nilai *pixel* yang sudah terkompresi dapat dilihat pada table dibawah ini :

Table 7. Nilai Desimal Terkompresi

Urutan Hexa Terkompresi	Nilai Desimal Terkompresi	Biner	Karakter
1	1	00000001	[]
2	33	00100001	!
3	35	00100011	#
4	69	01000101	E
5	103	01100111	G
6	128	10000000	€
7	129	10000001	
8	103	01100111	G
9	128	10000000	€
10	129	10000001	
11	4	00000100	[]

Berdasarkan hasil kompresi diatas dengan mengubah video menjadi kedalam table algoritma *start step stop code* sehingga didapat hasil :

00000001001000110100010001000100010001010100010001100111100000001000000 1

Ukuran *file* awal sebelum dikompesi adalah 128 bit yang di dapat dari perkalian antara bit dan frekuensi seperti yang ada pada tabel 3.5 di atas, sehingga rasio kompresinya adalah :
RC = (Ukuran *file* asli) / (Ukurang *file* terkompresi)
RC = 128/88 = 1,4

Jika dinyatakan dalam bentuk persen maka dituliskan dalam rumus sebagai berikut :
SS = (1 – (Ukuran *file* terkompresi) / (Ukuran *file* asli) x 100%
SS = (1- (88/128)) x 100%
SS = (1 – 0,6) x 100%
SS = 0,4 x 100%
SS = 40%

Dari perhitungan diatas, dapat disimpulkan bahwa dengan menggunakan algoritma *Start step stop code* karakter diatas dapat di kompresi sebanyak 40 %.

Proses dekompresi untuk *file* hasil kompresi menggunakan algoritma *start step stop code* dilakukan dengan menggunakan metode *Brute Force*. Pembacaan *string* bit dilakukan dari index terkecil sampai indeks terakhir dengan terus menambjakan nilai pada index sebelumnya yang tidak mewakili karakter *Start step stop code*. Mengembalikan *binary* menjadi *string* bit semula dengan menghilangkan biner yang ditebalkan seperti berikut:

000000010010001101000100010001000100010101000100011001111000**00010000001**

Untuk mengembalikan *binary* menjadi *string* bit semula dapat dilakukan melalui langkah berikut ini. Lakukan pembacaan pada 8 bit terakhir, hasil pembacaan berupa bilangan desimal. Nyatanya hasil pembacaan dengan n Hilangkan bit pada bagian akhir sebanyak 7 + n. setelah dilakukan perhitungan pembacaan bit akhir. Nilai biner yang dihilangkan sebanyak 8 bit pada akhir. n = 1. Hilangkan 7 + n atau 7 + 4 = 11 . Penjelasan diatas menunjukkan bahwa bit akhir harus dihilangkan. Hasil pengembalian *binary* menjadi *string* bit semula dapat dilihat sebagai berikut :

00000001001000110100010001000100010001010100010001100111100000

Proses dekompresi dilakukan dengan membaca string bit hasil kompresi *Start Step Stop Code* diatas berjumlah 77 bit seperti diawal sehingga dilakukan pembacaan string bit awal. Adapun pembacaan hasil perhitungan diatas adalah:

Tabel 8. Pengecekan Bit

Indeks	Nilai	Keterangan
1	0	Tidak ada
2	00	Tidak Ada
3	000	Tidak Ada
4	0000	Ada
5	0	Tidak Ada
6	00	Tidak Ada
7	001	Tidak Ada
8	0001	Ada
9	0	Tdak Ada
10	00	Tidak Ada
11	001	Tidak Ada
12	0010	Ada
13	0	Tidak Ada
14	00	Tidak Ada
15	001	Tidak Ada
16	0001	Ada
17	0	Tidak Ada
18	00	Tidak Ada
19	001	Tidak Ada
20	0010	Ada
21	0	Tidak Ada
22	00	Tidak Ada
23	001	Tidak Ada
24	0011	Ada
25	0	Tidak Ada
26	01	Tidak Ada
27	010	Tidak Ada
28	0100	Ada
29	0	Tidak Ada
30	01	Tidak Ada
31	010	Tidak Ada
32	0101	Ada
33	0	Tidak Ada
34	01	Tidak Ada
35	011	Tidak Ada

Indeks	Nilai	Keterangan
36	0110	Ada
37	0	Tidak Ada
38	01	Tidak Ada
39	011	Tidak Ada
40	0111	Ada
41	1	Tidak Ada
42	10	Tidak Ada
43	100	Tidak Ada
44	1000	Tidak Ada
45	100000	Tidak Ada
46	1000000	Tidak Ada
47	10000000	Ada
48	1	Tidak Ada
49	10	Tidak Ada
50	100	Tidak Ada
51	1000	Tidak Ada
52	100000	Tidak Ada
53	1000000	Tidak Ada
54	10000001	Ada
55	0	Tidak Ada
56	01	Tidak Ada
57	011	Tidak Ada
58	0110	Ada
59	0	Tidak Ada
60	01	Tidak Ada
61	011	Tidak Ada
62	0111	Ada
63	1	Tidak Ada
64	10	Tidak Ada
65	100	Tidak Ada
66	1000	Tidak Ada
67	100000	Tidak Ada
68	1000000	Tidak Ada
69	10000000	Ada
70	0	Tidak Ada
71	00	Tidak Ada
72	000	Tidak Ada
73	0001	Tidak Ada
74	00010	Tidak Ada
75	000100	Tidak Ada
76	0001000	Tidak Ada
77	00010001	Ada

Selanjutnya proses dekompresi dilakukan dengan algoritma *Start step stop code* yang dilakukan dengan mengubah hasil seluruh biner menjadi *string* bit semula yang terdapat pada *start step stop code file* awal.

Tabel 9. Nilai Start Step Stop Code Awal

No	Heksa	Biner
1	00	0000
2	00	0001
3	00	0010
4	20	0001
5	66	0010
6	74	0011
7	79	0100
8	70	0101
9	69	0110
10	73	0111
11	6F	10000000
12	6D	10000001
13	69	0110

No	Heksa	Biner
14	73	0111
15	6F	10000000
16	32	00010001

4. KESIMPULAN

Setelah melakukan penelitian pada kompresi *file* video berdasarkan algoritma *Start Stop Step Code* maka diperoleh kesimpulan tersebut Proses kompresi *file* video dengan menggunakan algoritma *Start Stop Step Code* telah berhasil dilakukan dengan *file* video yang berformat *mp4 dan proses kompresi yang dilakukan dapat berjalan sesuai dengn teknik kompresi. Penerapan algoritma *Start Stop Step Code* telah dilakukan dan terbukti bahwa *file* video yang memiliki ukuran besar dapat dikompres menjadi ukuran yang lebih kecil

REFERENCES

[1] S. Rizky, *Konsep Dasar Rekayasa Perangkat Lunak (Software Reengineering)*. Jakarta: Prestasi Pustaka, 2011.
[2] H. F. Siregar and M. Melani, "Perancangan Aplikasi Komik Hadist Berbasis Multimedia," *J. Teknol. Inf.*, vol. 2, no. 2, p. 113, 2019, doi: 10.36294/jurti.v2i2.425.
[3] R. D. Pratiwi, S. D. Nasution, and F. Fadlina, "PERANCANGAN APLIKASI KOMPRESI FILE TEKS DENGAN MENERAPKAN ALGORTIMA FIXED LENGTH BINARY ENCODING (FLBE)," *J. MEDIA Inform. BUDIDARMA*, vol. 2, no. 1, pp. 10–14, Jan. 2018, doi: 10.30865/mib.v2i1.813.
[4] K. Kunci and K. Zhao, "Vol 3 . No 2 . Desember 2014 VIDEO WATERMARKING UNTUK PERLINDUNGAN HAK CIPTA DENGAN," *J. Itsmart*, vol. 3, no. 2, pp. 52–65, 2014.
[5] J. H. Rompas, S. D. E. Paturusi, T. Elektro, U. Sam, and R. Manado, "Penerapan Video Mapping Multi Proyektor Untuk Mempromosikan Kabupaten Minahasa Selatan," vol. 14, no. 04, pp. 493–504, 1996.
[6] U. S. Utara, U. S. Utara, and U. S. Utara, "Analisis Perbandingan Kinerja Algoritma Start-Step-Stop Code dan Gopala-Hemachandra Code 2 (GH-2 (n)) pada Kompresi File Teks," vol. 2, 2019.
[7] Jeffery L. Whitten [et al], *Metode Desain & Analisis Sistem Edisi International*, Ed.ke-6., vol. Edisi 6. Jakarta: Andi, 2004.
[8] M. Djon Irwanto, S.Kom., *Perancangan Object Oriented Software Dengan UML*, I. Yogyakarta: Andi, 2007.
[9] R. A. S. and M. Shalahuddin, *Rekayasa Perangkat Lunak*. Bandung, 2016.
[10] A. P. Aditya, *Dasar-Dasar Pemrograman Database Dekstop Dengan Visual Basic.Net 2008*. Jakarta: PT. Elex Media Komputindo, 2013.
[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.