**Jurnal Sains Dan Teknologi Informasi**
Vol 1, No 2, Maret 2022, Hal 60-66
ISSN 2809-610X (Media Online)
DOI 10.47065/jussi.v1i2.1417
Website https://ejurnal.seminar-id.com/index.php/jussi

# Implementasi Self Delimiting Codes Untuk Kompresi File Teks

**Yesyua Fitri Rayani Marpaung**

Gram Studi Tenik Informatika, ProFakultas Ilmu Komputer Dan Teknologi Informasi, Universitas Budi Darma,
Jalan Sisingamangaraja No.338, Medan, Sumatera Utara, Indonesia
Email: fitrirayani31marpaung@gmail.com

**Abstrak**−Metode self delimiting codes adalah metode yang digunakan untuk mengkompres file indeks terbalik. Bilangan bulat yang di kompres positif kecil karena ada perbedaan dari pointer berurutan yang ada dalam urutan. Penggunaan algoritma self delimiting codes akan dilakukan berdasarkan karakter yang sering muncul dan akan memiliki jumlah bit terkecil berdasarkan kode self delimiting codes, sedangkan karakter yang paling sedikit muncul akan memiliki jumlah bit terpanjang. Kompresi data adalah proses pengkodean informasi dengan menggunakan bit yang lebih sedikit dibandingkan dengan kode yang sebelumnya dipakai dengan menggunakan skema pengkodean tertentu. Kompresi merupakan teknik memperkecil file yang berukuran besar dan mengurangi kebutuhan ruang penyimpanan. Kompresi data, terutama untuk komunikasi, dapat bekerja jika kedua pihak antara pengirim dan penerima data komunikasi memiliki skema pengkodean yang sama File teks merupakan file yang berisi informasi-informasi dalam bentuk teks. Data yang berasal dari dokumen pengolah kata, angka yang digunakan dalam perhitungan, nama dan alamat dalam basis data merupakan contoh masukan data teks yang terdiri dari karakter, angka dan tanda baca

**Kata Kunci:** Self Delimiting Codes, Kompresi ,File Teks.

**Abstract**−Self delimiting codes method is a method used to compress reverse index files. Positively compressed integers are small because there is a difference from the sequential pointers that are in the order. The use of self-delimiting codes algorithm will be based on the characters that appear frequently and will have the smallest number of bits based on the self-delimiting codes, while the characters that appear the least will have the longest number of bits. Data compression is the process of encoding information using fewer bits than the previously used code using a certain encoding scheme. Compression is a technique to reduce large files and reduce storage space requirements. Data compression, especially for communication, can work if both parties between the sender and receiver of communication data have the same encoding scheme. Text file is a file that contains information in text form. Data originating from word processing documents, numbers used in calculations, names and addresses in the database are examples of text data input consisting of characters, numbers and punctuation marks.

**Keywords:** Self Delimiting Codes, Compression, Text Files.

## 1. PENDAHULUAN

Saat ini perkembangan teknologi semakin meluas dengan kebutuhan masyarakat untuk memenuhi informasi dengan cepat. Berbagai macam fasilitas teknologi akan terus dikembangkan agar masyarakat dapat melakukan pertukaran informasi dalam bentuk teks, gambar, audio dan video dengan baik. Semakin tinggi permintaan informasi yang *real-time*, maka perpindahan data pun akan 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.

Saat ini perpindahan data sangat mudah dilakukan, tetapi yang menjadi kendala yang sangat mendasar adalah tempat penyimpanannya. Hal ini disebabkan karena ukuran data yang akan dipindahkan tidak sesuai dengan media penyimpanan yang tersedia. Dengan ukuran file yang semakin besar maka para pemakai computer menuntut untuk melakukan berbagai macam cara agar dapat menyimpan sejumlah file yang berukuran besar dalam media penyimpanan yang terbatas. Hal inilah yang menyebabkan file harus dimanfaatkan agar ukurannya akan menjadi lebih kecil. Teknik pemanfaatan data ini disebut dengan teknik kompresi data. Adapun tujuan kompresi data ialah untuk mengurangi ukuran file sebelum menyimpan dan memindahkan data ke dalam media penyimpanan. Pada umumnya teks berisi rangkaian karakter yang dapat membentuk suatu kata, surat dan narasi. Misalnya, file teks yang berekstensi (*.doc) yang ukurannya besar akan mengakibatkan proses pengiriman semakin lama, serta menggunakan ruang memori yang cukup besar dalam penyimpanannya. Maka untuk menghemat ruang penyimpanan dan mempercepat proses pengiriman file teks, perlu dilakukan proses kompresi agar ukuran file teks tersebut menjadi lebih kecil.

Teknik kompresi memiliki dua metode utama yaitu metode *lossless* dan metode *lossy*. Metode *lossy* adalah kompresi dimana terdapat data yang hilang selama proses kompresi yang mengakibatkan kualitas data yang hilang selama proses kompresi terjadi dihasilkan jauh lebih rendah dari kualitas data asli, sedangkan metode *lossless* tidak menghilangkan data selama proses kompresi terjadi, maka kualitas data hasil kompresi tidak menurun. Algoritma kompresi yang dapat digunakan adalah algoritma *self delimiting codes.*

Berdasarkan penelitian ini, algoritma yang digunakan untuk kompresi *file* teks adalah algoritma *self delimiting codes.* Algoritma *self delimiting codes* ini memiliki beberapa sifat dari dasar kode pemutakhiman yang di ambil dari referentasi baahasa parenthesis dari fungsi terbatas. Aspek yang paling menarik dari kode ini adalah sifat yang mirip fraktalat yang semua sub komponennya juga bersifat nyata[1].

Pada penelitian sebelumnya dapat disimpulkan hasil dari kompresi *file* teks menggunakan metode *self delimiting codes* mampu menghasilkan mengkompresi *file* teks dengan algoritma *self delimiting codes* yang akan menghasilkan sebuah *file* yang sudah terkompres, kemudian dalam mengembalikan *file* seperti semula akan melalui proses dekompresi[2].

**Jurnal Sains Dan Teknologi Informasi**
Vol 1, No 2, Maret 2022, Hal 60-66
ISSN 2809-610X (Media Online)
DOI 10.47065/jussi.v1i2.1417
Website https://ejurnal.seminar-id.com/index.php/jussi

.

# 2. METODOLOGI PENELITIAN

## 2.1 Tahapan Penelitian

Untuk mendukung kelancaran penelitian ini, maka dilakukan tahapan penelitian sebagai berikut:

a. Studi Literatur
   Pada tahap ini dilakukan pengumpulan referensi yang diperlukan dalam penelitian. Hal ini dilakukan untuk memperoleh informasi dan data yang diperlukan untuk penulisan skripsi ini. Referensi yang digunakan dapat berupa buku, jurnal, artikel, paper, makalah baik berupa media cetak maupun media internet mengenai kompresi *loseless* untuk file teks.
b. Analisis Sistem
   Pada tahap ini akan dianalisis sistem yang akan dibuat, batasan sistem, kinerja sistem dan cara kerja sistem. Sehingga sistem dapat menerapkan algoritma *self delimiting codes*.
c. Perancangan Sistem
   Merancang *input*, *output*, struktur file, program, prosedur, perangkat keras dan perangkat lunak yang diperlukan untuk medukung sistem informasi
d. Implementasi Sistem
   Sistem diimplementasikan dengan menggunakan Algoritma *self delimiting codes.*
e. Pengujian Sistem
   Pada tahap ini dilakukan pengujian kinerja sistem dan kebenaran hasil kompresi file teks yang dilakukan dengan algoritma *self delimiting codes.*
f. Dokumentasi Sistem
   Pada tahap ini seluruh kegiatan dalam pembuatan sistem didokumentasikan kedalam bentuk tulisan berupa laporan tugas akhir.

## 2.2 Algoritma *Skipjack*

Kompresi data adalah proses pengkodean informasi dengan menggunakan bit yang lebih sedikit dibandingkan dengan kode yang sebelumnya dipakai dengan menggunakan skema pengkodean tertentu[3]. Kompresi merupakan teknik memperkecil file yang berukuran besar dan mengurangi kebutuhan ruang penyimpanan. Kompresi data, terutama untuk komunikasi, dapat bekerja jika kedua pihak antara pengirim dan penerima data komunikasi memiliki skema pengkodean yang sama [4].

Berdasarkan hasil proses kompresi datayang dihasilkan dan pengembalian hasil kompresi ke data sebelum terkena kompresinya, jenis kompresi data dibedakan menjadi dua yaitu KompresiLoseless dan Kompresi Lossy.

a. Metode Loss less
   Kompresi lossless adalah kompresi data yang menghasilkan file data hasil kompresi yang dapat dikembalikan menjadi file data asli sebelum dikompresi secara utuh tanpa perubahan apapun. Kompresi jenis ini ideal untuk kompresi teks.Algoritma yang termasuk dalam metode kompresi lossless diantaranya adalah dictionary coding dan huffman coding.
b. Metode Lossy
   Kompresi data yang menghasilkan file data hasil kompresi yang tidak dapat dikembalikan menjadi file data sebelum dikompresi secara utuh. Ketika data hasil kompresi di-decode kembali, data hasil decoding tersebut tidak dapat dikembalikan menjadi sama dengan data asli tetapi ada bagian data yang hilang.

## 2.3 File Teks

File teks merupakan file yang berisi informasi-informasi dalam bentuk teks. Data yang berasal dari dokumen pengolah kata, angka yang digunakan dalam perhitungan, nama dan alamat dalam basis data merupakan contoh masukan data teks yang terdiri dari karakter, angka dan tanda baca[5].

## 2.4 Algoritma*Self Delimiting Codes*

Dalam matematika dan komputasi algoritma merupakan kumpulan perintah yang saling berkaitan untuk menyelesaikan suatu masalah.. Perintah-perintah ini dapat diterjemahkan secara bertahap dari awal hingga akhir. Dalam penyusunannya diperlukan urutan serta logika agar algoritma yang dihasilkan sesuai dengan yang diharapkan.Algoritma merupakan bagian terpenting yang tidak dapat dipisahkan dari pemrograman. Meskipun sintaksis dan semantik yang dibuat benar adanya, dengan algoritma yang keliru, permasalahan yang ingin dipecahkan dengan teknik pemrograman tidak akan berhasil. Oleh karena itu, sebelum membuat program aplikasi, hal pertama yang harus kita pahami algoritma atau prosedur pemecahannya. Hal ini bertujuan agar program yang telah dibuat dapat sesuai dengan yang diharapkan[6].

Untuk membangun self-delimiting kode, kode yang memiliki panjang variabel dan dapat diterjemahkan secara jelas[2]

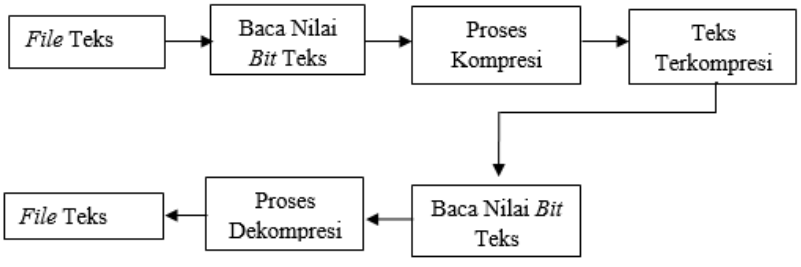
**Jurnal Sains Dan Teknologi Informasi**
Vol 1, No 2, Maret 2022, Hal 60-66
ISSN 2809-610X (Media Online)
DOI 10.47065/jussi.v1i2.1417
Website https://ejurnal.seminar-id.com/index.php/jussi

a. Gandakan setiap bit dari pesan asli, sehingga pesan menjadi satu set pasangan bit identik, lalu tambahkan sepasang bit yang berbeda. Dengan demikian, pesan 01001 menjadi bitstring 00 | 11 | 00 | 00 | 11 | 01. Ini sederhana tetapi jelas terlalu panjang. Itu juga rapuh, karena satu bit yang buruk akan membingungkan decoder (komputer atau manusia). Variasi dari teknik ini mendahului setiap bit angka dengan bit intercalary 1, kecuali bit terakhir, yang didahului dengan 0. Jadi, 01001 menjadi ¯10¯11¯10¯10¯01. Kita juga bisa konsentrasikan bit antar kalkulus bersama dan minta mereka diikuti oleh jumlahnya, seperti pada 11110 | 01001 (yang merupakan nomor itu sendiri didahului dengan kode unary-nya).
b. Siapkan tajuk dengan panjang pesan dan tambahkan ke pesan. Ukuran tajuk tergantung pada ukuran pesan, jadi tajuk seharusnya membuat self-delimiting menggunakan metode 1 di atas. Dengan demikian, pesan 6-bit 010011 menjadi tajuk 00 | 11 | 11 | 00 | 01 diikuti oleh 010011. Tampaknya hasilnya masih sangat panjang (16 bit untuk mengkodekan enam bit), tetapi ini karena pesan kami sangat singkat. Diberikan 1 juta pesan bit, panjangnya membutuhkan 20 bit. Header pembatas diri adalah 42 bit panjang, meningkatkan panjang pesan asli sebesar 0,0042%.
c. Jika pesannya sangat panjang (triliunan bit) sundulannya mungkin terlalu panjang. Dalam kasus seperti itu, kita dapat membuat header itu sendiri membatasi diri dengan menulisnya dalam format mentah dan sebelumnya dengan tajuknya sendiri, yang dibuat sendiri dengan metode 1.
d. Sekarang jelas bahwa mungkin ada sejumlah header. Header pertama adalah membuat *self-delimiting* dengan metode 1, dan semua header lainnya digabungkan ke dalamnya format mentah. Komponen terakhir adalah pesan biner asli (sangat panjang).
e. Integer panjang variabel desimal dapat direpresentasikan dalam basis 15 (quindecimal) sebagai serangkaian nibble (masing-masing kelompok terdiri dari empat bit), dimana setiap nibble adalah basis-15 digit(mis., antara 0 dan 14) dan nibble terakhir mengandung 16 = 11112. Metode ini terkadang disebut sebagai kode nibble atau byte coding. Tabel 3.25 mencantumkan beberapa contoh.
f. Variasi pada kode nibble dimulai dengan representasi biner dari integer n (atau n - 1), tambahkan dengan nol sampai jumlah bit habis dibagi 3, pisahkan ke dalam kelompok masing-masing tiga bit, dan awali setiap grup dengan 0, kecuali kelompok paling kiri (atau sebagai alternatif, paling kanan), yang diawali dengan 1. Panjang kode ini untuk bilangan bulat n adalah 4 (log2 n) / 3, sehingga sangat ideal untuk distribusi formulir. Ini adalah distribusi kuasa hukum dengan parameter 3/4. Perpanjangan alami dari ini kodeuntuk grup k-bit. Kode semacam itu cocok dengan distribusi kuasa hukum formulir. Ini adalah rumus distribusi kuasa
g. Jika data yang akan dikompresi terdiri dari sejumlah besar bilangan bulat positif kecil, maka skema pengemasan yang selaras kata mungkin memberikan kompresi yang baik (meskipun bukan yang terbaik) dikombinasikan dengan decoding cepat. Idenya adalah untuk mengemas beberapa bilangan bulat menjadi panjang tetap bidang kata komputer. Jadi, jika ukuran kata adalah 32 bit, 28 bit dapat dipartisi menjadi beberapa bidang k-bit sedangkan empat bit sisanya menjadi pemilih yang menunjukkan nilai k.

## 3. HASIL DAN PEMBAHASAN

Berdasarkan penelitian ini, akan dilakukan analisa dan perancangan perangkat lunak pengkompresian *file* teks dengan menggunakan algoritma *Self Delimiting codes*. Algoritma *self delimiting codes* merupakan salah satu teknik kompresi lossless yang dapat memperkecil suatu data berdasarkan dengan frekuensi karakter pada objek yang akan dilakukan proses kompresi. Penggunaan algoritma *self delimiting codes* akan dilakukan berdasarkan karakter yang sering muncul dan akan memiliki jumlah bit terkecil berdasarkan kode *self delimiting codes*, sedangkan karakter yang paling sedikit muncul akan memiliki jumlah bit terpanjang. Tahapan analisa terhadap suatu sistem dilakukan sebelum tahapan perancangan dilakukan.

*File* teks yang semakin besar ukuran datanya maka semakin besar pula tempat penyimpanannya, dan proses transmisi yang dibutuhkan juga semakin besar. Dalam melakukan proses kompresi *file* teks sebelumnya harus dilakukan analisa terhadap data teks. Untuk mengetahui prosedur kompresi dan dekompresi suatu *file* teks dapat dilihat pada Gambar 1.

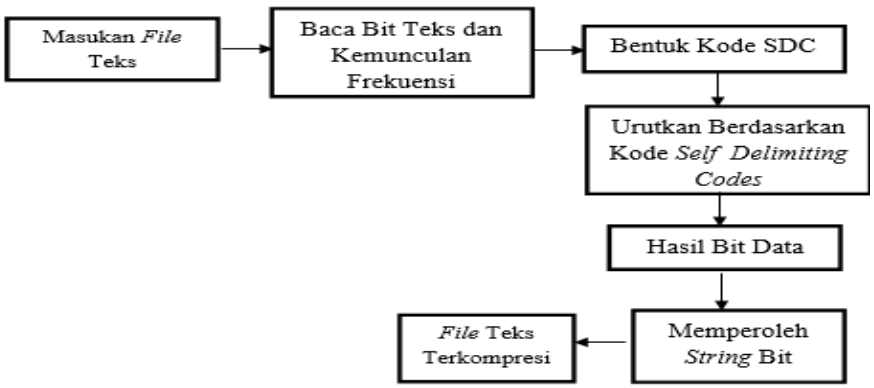



**Gambar 1.** Prosedur Kompresi Dekompresi *File* Teks

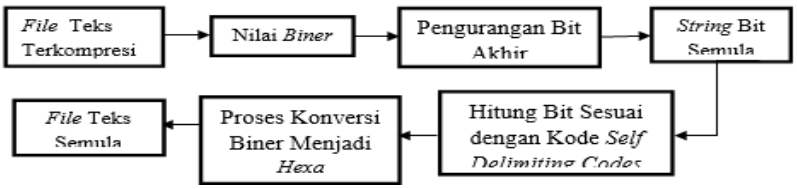
Dalam sistem pengkompresian *file* teks, metode yang digunakan pada sistem ini adalah dengan menggunakan algoritma *self delimiting codes*. Sub bab ini menjelaskan tentang tahapan atau langkah-langkah dari proses algoritma yang

**Jurnal Sains Dan Teknologi Informasi**
Vol 1, No 2, Maret 2022, Hal 60-66
ISSN 2809-610X (Media Online)
DOI 10.47065/jussi.v1i2.1417
Website https://ejurnal.seminar-id.com/index.php/jussi

digunakan. Tahapan pertama yang dilakukan yaitu proses pengkompresian dan tahap kedua yaitu proses dekompresi terhadap datateks yang telah terkompresi. Adapun penyusunan tahapan algoritma dapat dilihat pada Gambar dibawah ini.




**Gambar 2.** Tahapan Algoritma Terkompresi




**Gambar 3.** Tahapan Algoritma Dekompresi

Dalam penelitian ini, penulis akan membahas 2 proses utama yaitu proses kompresi dan proses dekompresi, penulis akan mengkompresi sebuah *file* teks dengan menggunakan algoritma *self delimiting codes* merupakan salah satu algoritma yang termasuk di dalam metode *lossless* data. Sebelum *file* teks dikompresi, terlebih dahulu dilakukan pembacaan biner yang terdapat pada *file* teks untuk mendapatkan data berupa data biner. Membaca biner yang terdapat pada *file* teks menggunakan table ASCII untuk mencari nilai biner pada *file* teks. Berikut adalah contoh *file* teks yang akan di kompresi dan didekompresi.

Dalam penerapan ini digunakan nama penulis sendiri sebagai *file* teks berisi string "YESYUA FITRI". Untuk mengetahui ukuran string tersebut, dapat dilihat pada Tabel 1.

**Tabel 1.** *File* yang belum dikompresi

| Karakter | *Hexa* Desimal | Ascii *Biner* | Bit |
|---|---|---|---|
| Y | 59 | 01011001 | 8 |
| E | 45 | 01000101 | 8 |
| S | 53 | 01010011 | 8 |
| Y | 59 | 01011001 | 8 |
| U | 55 | 01010101 | 8 |
| A | 41 | 01000001 | 8 |
| *(space)* | 20 | 00100000 | 8 |
| F | 46 | 01000110 | 8 |
| I | 49 | 01001001 | 8 |
| T | 54 | 01010100 | 8 |
| R | 52 | 01010010 | 8 |
| I | 49 | 01001001 | 8 |
| | Jumlah Bit | | 96 |

Setelah *file* teks berhasil dibaca menjadi *string*, maka tahap selanjutnya adalah proses pemasukan *file* teks kedalam tabel untuk dilakukan pembacaan frekuensi. Pembacaan frekuensi dilakukan dengan menghitung jumlah nilai yang sama di setiap teks yang muncul. Adapun pembacaan frekuensi dapat dilihat pada tabel di dibawah ini.

**Tabel 2.** Pendataan Simbol

| No | Karakter | Frekuensi |
|---|---|---|
| 1 | Y | 2 |
| 2 | I | 2 |
| 3 | E | 1 |
| 4 | S | 1 |
| 5 | U | 1 |

**Jurnal Sains Dan Teknologi Informasi**
Vol 1, No 2, Maret 2022, Hal 60-66
ISSN 2809-610X (Media Online)
DOI 10.47065/jussi.v1i2.1417
Website https://ejurnal.seminar-id.com/index.php/jussi

| | | |
|---|---|---|
| 6 | A | 1 |
| 7 | (*space)* | 1 |
| 8 | F | 1 |
| 9 | T | 1 |
| 10 | R | 1 |
| Jumlah | | 12 |

Berdasarkan pada tabel di atas, didapatkan beberapa karakter teks yang sama. Sebelum proses kompresi , langkah awal adalah membaca karakter teks kemudian membuat tabel nilai *heksa* yang diurutkan dari nilai frekuensi terbesar (nilai *heksa* yang sama) ke terkecil. Urutan nilai *hexa* dapat dilihat pada tabel di bawah ini

**Tabel 3.** Hexa Yang Belum Dikompresi

| No | Karakter | *Hexa decimal* | *Biner* | Frekuensi | Bit | Frekuensi x Bit |
|---|---|---|---|---|---|---|
| 1 | Y | 59 | 01011001 | 2 | 8 | 16 |
| 2 | I | 49 | 01001001 | 2 | 8 | 16 |
| 3 | E | 45 | 01000101 | 1 | 8 | 8 |
| 4 | S | 53 | 01010011 | 1 | 8 | 8 |
| 5 | U | 55 | 01010101 | 1 | 8 | 8 |
| 6 | A | 41 | 01000001 | 1 | 8 | 8 |
| 7 | (*space)* | 20 | 01000000 | 1 | 8 | 8 |
| 8 | F | 46 | 01000110 | 1 | 8 | 8 |
| 9 | T | 54 | 01010100 | 1 | 8 | 8 |
| 10 | R | 52 | 01010010 | 1 | 8 | 8 |
| Jumlah | | | | | | 96 |

Berdasarkan tabel diatas, satu nilai *decimal* (karakter) bernilai 8 bit bilangan biner. Sehingga 12 bilangan *decimal* mempunyai nilai biner sebanyak 96 bit. Untuk mengubah satuan menjadi *byte* maka jumlah keseluruhan bit dibagikan 8, maka dihasilkan 96/8 = 12 *byte*. Langkah selanjutnya adalah dengan membentuk tabel kode *self delimiting codes.* Aturan dalam pembentukan kode bilangan dengan menggunakan kode *self delimiting codes* dapat dilihat pada sub landasan teori bab sebelumnya. Adapun kode *self delimiting codes*.dapat dilihat pada tabel 4. di bawah ini:

**Tabel 4.** kode *Self Delimiting Codes*

| Index | Jumlah Kode | Panjang kode | Bit yang tidak digunakan |
|---|---|---|---|
| 0 | 96 | 1 | 0 |
| 1 | 48 | 2 | 0 |
| 2 | 32 | 3 | 0 |
| 3 | 24 | 4 | 0 |
| 4 | 19 | 5 | 1 |
| 5 | 16 | 6 | 0 |
| 6 | 12 | 8 | 0 |

Proses selanjutnya melakukan kompresi bilangan biner berdasarkan tabel kode *self delimiting code* .Proses kompresi algoritma dilakukan dengan membagi 8 bit, diketahui bit pertama adalah 01011001 lakukan pengurangan bit berdasarkan tabel algoritma *self delimiting codes* yang dicari. Adapun hasilnya sebagai berikut.
01011001 → panjang kode 2 dengan bit yang tidak digunakan adalah 0 maka *code word* nya adalah 010111. Maka hasil pengurangan bit berdasarkan tabel algoritma *self delimiting codes* dapat dilihat pada table 5 dibawah ini

**Tabel 5.** Kompresi Bilangan Biner *Self Delimiting codes*

| No | Karakter | Hexa decimal | Biner | Panjang kode SDC | Bit yang tidak digunakan | *Code Word* |
|---|---|---|---|---|---|---|
| 1 | Y | 59 | 01011001 | 2 | 0 | 010111 |
| 2 | I | 49 | 01001001 | 2 | 0 | 0111 |
| 3 | E | 45 | 01000101 | 3 | 0 | 01101 |
| 4 | S | 53 | 01010011 | 2 | 0 | 010111 |
| 5 | U | 55 | 01010101 | 1 | 0 | 1111 |
| 6 | A | 41 | 01000001 | - | - | 01000001 |
| 7 | (*space)* | 20 | 01000000 | 6 | 0 | 01 |
| 8 | F | 46 | 01000110 | 3 | 0 | 01110 |
| 9 | T | 54 | 01010100 | 2 | 0 | 010101 |
| 10 | R | 52 | 01010010 | 2 | 0 | 010110 |
| Jumlah | | | | | | 11001100 |

Proses selanjutnya adalah melakukan kompresi nilai dari teks *hexa* sampel dengan nilai kode *self delimiting codes* yang di dapat dari tabel 5 di atas. Adapun Proses kompresi teks sampel dapat dilihat pada tabel di bawah ini :

**Jurnal Sains Dan Teknologi Informasi**
Vol 1, No 2, Maret 2022, Hal 60-66
ISSN 2809-610X (Media Online)
DOI 10.47065/jussi.v1i2.1417
Website https://ejurnal.seminar-id.com/index.php/jussi

Tabel 6. Nilai Hexa Yang Sudah Dikompresi Dengan Kode *Self Delimiting Codes*

| No | *Hexa decimal* | *Biner* | Panjang kode SDC | Bit yang tidak digunakan | *Code Word* |
|---|---|---|---|---|---|
| 1 | 59 | 01011001 | 2 | 0 | 010111 |
| 2 | 49 | 01001001 | 2 | 0 | 0111 |
| 3 | 45 | 01000101 | 3 | 0 | 01101 |
| 4 | 53 | 01010011 | 2 | 0 | 010111 |
| 5 | 55 | 01010101 | 1 | 0 | 1111 |
| 6 | 41 | 01000001 | - | - | 01000001 |
| 7 | 20 | 01000000 | 6 | 0 | 01 |
| 8 | 46 | 01000110 | 3 | 0 | 01110 |
| 9 | 54 | 01010100 | 2 | 0 | 010101 |
| 10 | 52 | 01010010 | 2 | 0 | 010110 |
| | | | Jumlah digit *biner* | | 52 |

Dapat dibentuk nilai bit baru hasil kompresi dari susunan nilai karakter sampel awal sebelum kompresi yaitu, YESYUA FITRI menjadi nilai bit *biner*.
"01011101101011011111010000010101110011101010101010110".

Berdasarkan penambahan bit maka jumlah bit setelah dikompresi adalah 52 bit, jika diubah menjadi *byte* maka 52/8 = 6,5 *byte*. Maka dapat disimpulkan bahwa ukuran *sampel* teks sebelum dikompresi adalah 12 *byte*, sedangkan ukuran *sampel* teks setelah dikompresi adalah 6,5 *byte*. Setelah pembagian dilakukan, maka *string bit* yang sudah dibagi dirubah kedalam suatu karakter dengan terlebih dahulu mencari nilai *hexa* dari *string bit* tersebut menggunakan kode ASCII untuk megetahui karakter dari *hexa* yang sudah terkompresi.

**Tabel 7.** Nilai *Hexadecimal* Terkompresi

| *Codeword* | Nilai *Hexadecimal* terkompresi | Karakter |
|---|---|---|
| 010111 | 17 | |
| 0111 | 7 | |
| 01101 | D | |
| 010111 | 17 | |
| 1111 | F | |
| 01000001 | 21 | ! |
| 01 | 1 | |
| 01110 | E | |
| 010101 | 15 | |
| 010110 | 16 | |

Setelah nilai *Hexal* diketahui, maka mengubah nilai *hexal* kedalam suatu karakter.

Dari hasil kompresi tersebut dapat diukur kinerja algoritma *Self Delimiting Codes* sebagai berikut :

a. *Ratio of Compression* ($R_C$)
Ratio of Compression ($R_C$) adalah nilai perbandingan antara ukuran bit data sebelum dikompresi dengan ukuran bit data yang telahdikompresi.

$$Rc = \frac{\text{Jumlah bit sebelum dikompresi}}{\text{Jumlah bit sesudah dikompresi}} \times 100\%$$

$$Rc = \frac{96}{52} = 1{,}8$$

b. *Compression Ratio* ($C_R$)
*Compression Ratio*($C_R$)adalah persentase perbandingan antara data yang sudah dikompresi dengan data yang belum dikompresi.

$$CR = \frac{\text{Ukuran data sesudah dikompresi}}{\text{Ukuran data sebelum dikompresi}} \times 100\%$$

$$C_R = \frac{6{,}5}{12} \times 100\% = 0{,}54$$

c. *Space Savings* ($S_S$)
Space Saving (SS) selisih antara data yang belum dikompresi dengan besar data yang dikompresi.
$S_S$ = 100 % - $C_R$
$S_S$ = 100% - 0,54% = 0,9946%

d. *Time Compression*
*Time compression* adalah waktu yang dibutuhkan untuk melakukan proses kompresi dan dekompresi. Semakin kecil waktu yang diperoleh maka semakin efisien metode yang digunakan dalam proses kompresi dan dekompresi itu.

Selanjutnya proses dekompresi yang dilakukan adalah menganalisa keseluruhan bit hasil dari kompresi sebelumnya. Adapun bit keseluruhan hasil kompresi dapat dilihat pada tabel berikut :

**Jurnal Sains Dan Teknologi Informasi**
Vol 1, No 2, Maret 2022, Hal 60-66
ISSN 2809-610X (Media Online)
DOI 10.47065/jussi.v1i2.1417
Website https://ejurnal.seminar-id.com/index.php/jussi

**Tabel 8.** Nilai *Hexadecimal* Terkompresi

| *Codeword* | Nilai *Hexadecimal* terkompresi | Karakter |
|---|---|---|
| 010111 | 17 | |
| 0111 | 7 | |
| 01101 | D | |
| 010111 | 17 | |
| 1111 | F | |
| 01000001 | 21 | ! |
| 01 | 1 | |
| 01110 | E | |
| 010101 | 15 | |
| 010110 | 16 | |

Selanjutnya proses dekompresi *self delimiting codes* yang akan dilakukan mengubah hasil seluruh biner menjadi karakter yang terdapat pada *file* awal, yaitu **: !**

Selanjutnya proses dekompresi dilakukan dengan algoritma *self delimiting codes* yang dilakukan dengan mengubah hasil seluruh *biner* menjadi *string bit* semula yang terdapat pada *file* awal.

**Tabel 9.** Dekompresi

| No | *CodeWord* | Pengembalian Kode | Karakter |
|---|---|---|---|
| 1. | 010111 | 2→0 | 01011001→ Y |
| 2. | 0111 | 2, 2 →0 | 01001001→I |
| 3. | 01101 | 3→0 | 01000101→E |
| 4. | 010111 | 2→0 | 01010011→S |
| 5. | 1111 | 1→0 | 01010101→U |
| 6. | 01000001 | - | 01000001→A |
| 7. | 01 | 6→0 | 01000000→*space* |
| 8. | 01110 | 3→0 | 01000110→F |
| 9. | 010101 | 2→0 | 01010100→T |
| 10. | 010110 | 2→0 | 01010010→R |

Hasil string bit semula:
"0101100101000101010100110101100101010101010000010010000001000110010010010101010001010010010010 01"
"YESYUA FITRI"

## 4. KESIMPULAN

Kesimpulan yang dapat diambil setelah melakukan hasil dan pembahasan dari kompresi file teks dengan menerapkan algoritma Self Delimiting Codes. Prosedur dalam mengkompresi sebuah file teks pada self delimitng codes ialah akan dilakukan berdasarkan karakter yang sering muncul dan akan memiliki jumlah bit terkecil berdasarkan kode self delimitng codes sedangkan karakter yang paling sedikit muncul akan memiliki jumlah bit terpanjang. Berdasakan hasil penerapan metode terhadap sampel data dimana nilai *Compression Ratio* sebesar 54%

## REFERENCES

[1] P. Tarau, "Hereditarily finite representations of natural numbers and self-delimiting codes," Proc. ACM SIGPLAN Int. Conf. Funct. Program. ICFP, pp. 11–17, 2010, doi: 10.1145/1863597.1863602.
[2] M. B. Nugroho, Handbook of Data Compression, vol. 53, no. 9. 2013.
[3] A. Muhammad dan A. Soeb, "Penarapan Algoritma Prefix Code Dalam Kompresi File Video," dalam KOMIK (Konferensi Nasional Teknologi Informasi dan Komputer), Medan, 2021.
[4] A. Nur dan A. Soeb, "Penerapan Algoritma Elias Omega Code Pada Kompresi File Audio Aplikasi Murottal Muzzamil Hasbalah," Pelita Informatika: Informasi dan Informatika, vol. 9, no. 2, pp. 113-119, 2020.
[5] F. Razi, D. I. Komputer, F. Matematika, D. A. N. Ilmu, P. Alam, and U. S. Utara, "Fahrur Razi : Analisis Pengaruh Panjang Bit Kode Pada Kinerja Program Kompresi Yang Menggunakan Algoritma Lempel Ziv Welch (LZW), 2009. USU Repository © 2009," 2009.
[6] D. A. Yansyah, "Perbandingan Metode Punctured Elias Code Dan Huffman Pada Kompresi File Text," J. Ris. Komput., vol. 2, no. 6, pp. 33–36, 2015.
[7] Y. Darnita, K. Khairunnisyah, and H. Mubarak, "Kompresi Data Teks Dengan Menggunakan Algoritma Sequitur," Sistemasi, vol. 8, no. 1, p. 104, 2019, doi: 10.32520/stmsi.v8i1.429.
[8] Sugiyono, "Manajemen Pengetahuan Sistem Informasi Pegawai Pt Guna Karya Indonesia ( Gki ) Bekasi," CKI SPOT, vol. 10, no. 2, pp. 35–46, 2017.