Journal Global Tecnology Computer
Vol 2, No. 2, April, 2023, Hal 42 - 53
ISSN: **2809-6118** (Online - Elektronik)
Website https://ejurnal.seminar-id.com/index.php/jogtc

Perbandingan Algoritma Tunstall Code Dengan Phased-In Code Pada Kompresi Pesan Teks Dengan Metode Single Exponential

Kristina Pakpahan

Program Studi Teknik Informatika, Fakultas Ilmu Komputer & Teknologi Informasi, Universitas Budi Darma, Medan, Indonesia
Jl. Sisingamangaraja No.338, Siti Rejo I, Kec. Medan Kota, Kota Medan, Sumatera Utara, Indonesia
Email: Kristina.pakpahan1357@gmail.com

Abstrak−Teks pada umumnya berisi rangkaian karakter dan dapat membentuk suatu kata, surat dan narasi. Misalnya, pesan teks 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 pesan teks, perlu dilakukan proses kompresi agar ukuran pesan teks tersebut menjadi lebih kecil. Teknik kompresi yang dapat memampatkan pesan teks adalah kompresi pesan menggunakan algoritma Tunstall code dan Phased-in code. Kedua algoritma ini memiliki perbedaan dalam mengompresi pesan teks. Sistem kerja algoritma Tunstall code adalah menentukan rangkaian/sequence simbol untuk setiap codeword, mengambil probabilitas tertinggi dan melakukan iterasi sebanyak N (jumlah symbol) yang akan dikompresi, sedangkan sistem kerja algoritma Phased-in code adalah memberikan sebuah alphabet untuk N simbol, dimulai dengan sebuah tabel kode yang terdiri dari simbol-simbol. Dari perbedaan algoritma tersebut, maka penulis membandingkan kedua algoritma dengan menggunakan metode perbandingan, yaitu metode Single exponential. Dengan menggunakan metode Single exponential, penulis berhasil membandingkan kedua algoritma, dan menghasilkan satu algoritma yang baik untuk pengompresian pesan teks, yaitu algoritma Tunstall code.

Kata Kunci: Kompresi, Pesan Teks, Algoritma *Tunstall Code*, Algoritma *Phased-in Code*, Metode *Single Exponential*

Abstract−*Text generally contains a series of characters and can form a word, letter and narrative. For example, a text message that is large in size causes the sending process to take longer, and uses a large memory space in its storage. So to save storage space and speed up the process of sending text messages, it is necessary to do a compression process so that the size of the text message becomes smaller. A compression technique that can compress text messages is message compression using the Tunstall code and Phased-in code algorithms. Both of these algorithms have differences in compressing text messages. The working system of the Tunstall code algorithm is to determine the sequence of symbols for each codeword, take the highest probability and perform N iterations (number of symbols) to be compressed, while the working system of the Phased-in code algorithm is to provide an alphabet for N symbols, starting with a code table consisting of symbols. From the differences in these algorithms, the authors compare the two algorithms using a comparison method, namely the Single exponential method. By using the Single exponential method, the authors managed to compare the two algorithms, and produced a good algorithm for compressing text messages, namely the Tunstall code algorithm.*

Keywords: *Compression, Text Message, Tunstall Code Algorithm, Phased-in Code Algorithm, Single Exponential Method*

1. PENDAHULUAN

Kemajuan teknlogi yang begitu pesat memicu kebutuhan informasi yang sangat besar. Hal ini menyebabkan dunia membutuhkan suatu metode yang dapat digunakan untuk menyebarkan informasi secara cepat, dan tidak memakai memori terlalu besar. Pesan teks atau SMS (*Short Message Service*) adalah sebuah layanan yang digunakan dalam telpon genggam untuk mengirim dan menerima pesan-pesan pendek [1]. Dalam melakukan pengiriman pesan teks, seorang pengguna dapat mengirim pesan lebih dari 140 karakter, dan pengguna harus membayar lebih dari sekali. Hal ini terjadi karena pesan yang dikirimkan terdiri lebih dari satu halaman sehingga proses pengiriman pesan akan dilakukan sebanyak jumlah halaman yang ada, jumlah halaman sesuai dengan isi SMS yang diketikkan. Salah satu solusi dari permasalahan tersebut adalah dengan melakukan kompresi pada pesan teks yang dikirimkan. Kompresi berarti mengecilkan suatu ukuran atau disebut juga dengan memampatkan. Kompresi data adalah proses mengkodekan informasi menggunakan bit yang lain dan lebih rendah sehingga hanya membutuhkan ruang penyimpanan lebih sedikit [2].

Banyak algoritma kompresi yang dapat digunakan, diantaranya, algoritma *tunstal code* dan *phased-in code*. Penggunaan algoritma *tunstal code* dan *phased-in code* ini bisa berarti pengirim mempersingkat panjang kata dengan singkatan-singkatan yang lazim digunakan. Solusi lain yang bisa digunakan. dengan melakukan proses *encoding* terhadap pesan singkat yang dikirim menggunakan jumlah bit yang lebih sedikit namun memiliki informasi yang sama, kemudian pesan yang telah di-*encoding* itu dikirimkan dan dibaca oleh ponsel penerima dengan menggunakan proses *decoding* sehingga tampilan pesan akan sama dengan yang dimaksud.

Setelah melakukan kompresi pesan teks dengan menggunakan algoritma *tunstal code* dan *phased-in code*, kemudian dilakukan perbandingan antara dua algoritma tersebut dengan menggunakan metode *Single Exponential*. Manfaat dalam membandingkan kedua algoritma tersebut adalah untuk mengetahui algoritma mana lebih efisien dalam mengkompresi pesan teks. Berdasarkan penelitian terdahulu yang dilakukan oleh Shmuel T.Klein bahwa algoritma klasik *tunstall code* mendesain kode panjang variabel optimal untuk yang diberikan distribusi frekuensi elemen yang akan dikodekan. Aspek-aspek lain dari kode panjang variabel bukan hanya rasio kompresi mereka, mungkin ada

insentif untuk kembali ke panjang tetap kode *decoding*, misalnya lebih rumit dengan panjang variabel, sebagai akhirnya dari setiap kode harus ditentukan. Panjang kode variabel yang dibawa biasanya juga memproses penalti waktu, terutama untuk decoding dan juga untuk fitur yang diinginkan lainnya, seperti kemungkinan untuk mencari langsung di dalam teks terkompresi, tanpa perlu untuk dekompresi terlebih dahulu [3].

Penelitian yang juga dilakukan oleh Yo-Sung Ho bahwa pada algoritma *phased-in code* berisi *codeword* dari berbagai panjang dan cocok untuk kompresi data di mana simbol individu memiliki probabilitas yang sangat berbeda. Kode-kode bagian ini juga disebut kode biner bertahap. Satu set berbasis-fase terdiri dari kode-kode pendek dari kekuatan-kekuatan dan mungkin memberikan kontribusi sesuatu (walaupun tidak banyak) pada kompresi data di mana simbol-simbol memiliki probabilitas yang kira-kira sama [4].

2. METODOLOGI PENELITIAN

2.1 Tahapan Penelitian

Dalam penelitian ini, ada beberapa tahapan yang akan dilakukan adalah sebagai berikut:

a. Studi Literatur
   Pada tahap ini penulis mengumpulkan referensi yang diperlukan dalam penelitian, yang mana dilakukan untuk memperoleh data-data atau informasi yang dibutuhkan dalam pembuatan penelitian ini. Pada tahap studi literatur dilakukan pengumpulan buku, jurnal, *e-book*, artikel, makalah, maupun situs internet yang membahas algortima *tunstal code, phased-in code,* dan metode *single exponential* untuk dipelajari lebih lanjut.
b. Analisis dan Perancangan Sistem
   Pada tahap analisis dan perancangan sistem akan dilaksanakan perancangan *flowchart*, *interface*, UML dengan menerapkan algoritma *tunstal code,phased- in code,* dan metode *single exponential.*
c. Perancangan Sistem
   Merancang *input*, *output*, struktur file, program, prosedur, perangkat keras dan perangkat lunak yang diperlukan untuk medukung sistem informasi
d. Implementasi Sistem
   Pada tahap implementasi sistem ini dilakukan implementasi terhadap hasil perancangan dengan cara melakukan penulisan program.
e. Pengujian Sistem
   Pengujian sistem yang sudah dikembangkan dengan sistematika yang sudah dirancang sedemikian rupa untuk melihat perangkat lunak memberikan hasilyang diinginkan.
f. Dokumentasi
   Dalam tahap dokumentasi dilakukan penyusunan laporan dari hasil perancangan aplikasi dalam format penulisan penelitian.

2.2 Perbandingan Algoritma

Sjahcran Basah mengatakan bahwa perbandingan merupakan suatu metode pengkajian atau penyelidikan dengan mengadakan perbandingan di antara dua objek kajian atau lebih untuk menambah dan memperdalam pengetahuan tentang objek yang dikaji. Jadi di dalam perbandingan ini terdapat objek yang hendak diperbandingkan yang sudah diketahui sebelumnya, akan tetapi pengetahuan ini belum tegas dan jelas [5]. Algoritma adalah sekumpulan langkah yang rinci yang ditujukan untuk menyelesaikan suatu masalah. Langkah-langkah yang dimaksud adalah agar bisa dituangkan kedalam program sehingga bisa dieksekusi oleh sistem computer. Namun, tanpa harus menuangkan ke dalam program, langkah-langkah yang terdapat di algoritma bisa diuji secara manual [6]. Suatu prosedur yang jelas untuk menyelesaikan suatu persoalan dengan menggunakan langkah-langkah tertentu dan terbatas jumlahnya. Susunan langkah yang pasti, yang bila diikuti maka akan mentransformasi data input menjadi output yang berupa informasi [7].

2.3 *Tunstall Code*

Tunstall code merupakan salah satu algoritma yang termasuk di dalam metode *lossless* data. Langkah pertama di dalam algoritma ini adalah membuat tabel simbol, frekuensi dan kolom *probability*. Setelah itu membuat simbol yang pendek ke dalam kemungkinan yang banyak. Kemudian melakukan iterasi untuk mengetahui berapa banyak iterasi yang dilakukan untuk dimasukkan ke dalam formula $N + k (N-1) \leq 2^n$. Untuk melakukan iterasi, simbol yang pendek menjadi probabilitas yang banyak. Kemudian menghapus simbol dengan probabilitas tertinggi, setelah itu masukan simbol dengan simbol yang terdapat di dalam tabel[8].

Ilustrasi dari algoritma ini adalah sebuah kalimat yang terdiri dari dua simbol A dan B dimana A lebih sering dijumpai. Berikan sebuah *string* yang unik dari kalimat ini, diberikan sub-*string* dalam *form* AA, AAA, AB, AAB dan B, tetapi *string* yang jarang dalam bentuk BB. Masukkan ukuran kode yang sudah dimodifikasi ke dalam 5 bentuk *string* yang sudah dibuat. AA = 000, AAA = 001, AB = 010, ABA = 011 dan B = 100. *Tunstall code* memberikan sebuah alphabet untuk N simbol, dimulai dengan sebuah tabel kode yang terdiri dari simbol-simbol. Kemudian akan

diiterasi sepanjang ukuran tabel kode yang lebih sedikit dari jumlah angka di dalam kode. Tiap iterasi mengikuti langkah-langkah sebagai berikut:

a. Membuat table simbol, frekuensi sebagai langkah pertama untuk mengetahui iterasi yang akan dilakukan.
b. Menghitung jumlah iterasi yang akan dilakukan dengan menggunakan rumus $N + k (N-1) \leq 2^n$
c. Membuat tabel simbol dan probabilitas untuk melakukan iterasi, untuk mencari probabilitas dengan rumus F/N.
d. Pilih simbol dengan probabilitas terbanyak di dalam tabel, dan menggabungkannya dengan karakter lainnya.
e. Mengubah teks menjadi kode di dalam *Tunstall code* [8].

2.4 *Phased-In Code*

Phased-in code adalah salah satu algoritma yang memiliki simbol yang sama dan tidak dapat dikompresi oleh peangkat lunak pemutar beraga berkas media dan biasanya diberikan kode panjang tetap. Kode-kode dalam algoritma ini biasanya disebut kode biner bertahap. Satu bagian algoritma ini terdiri dari kode-kode pendek yang memiliki kekuatan dan dapat dikumpulkan untuk kompresi data dengan simbol yang memiliki probabilitas yang kemungkinan sama. Banyak kode prefix yang dijelaskan disini yang dikembangkan untuk kompresi tipe data tertentu. Kode ini biasanya berisi *codeword* dan cocok untuk kompresi data dimana setiap simbol itu memiliki kemungkinan yang berbeda. Langkah pertama di dalam algoritma ini adalah membuat n data simbol, dimana $n = 2^m$ (menunujukkan bahwa $m = \log_2 n$), dengan menetapkan kode m-bit sebagai *codeword* .Namun, jika $2^{m-1} < n < 2^m$, maka $= \log_2 n$ bukan bilangan bulat.

Jika kode panjang ditetapkan pada simbol, setiap *codeword* mempunyai panjang yang tetap sesuai simbol, setiap *codeword* mempunyai panjang $[\log_2 n]$ bit, tetapi tidk semu *codeword* digunakan. Contoh nya n =1.000, dalam hal ini setiap *codeword* dengan panjang tetap adalah $[\log_2 1000] = 10$ bit, tetapi hanya 1.000 dari 1.024 *codeword* kemungkinan yang digunakan. Dalam pendekatan yang dijelaskan disini, terdapat 2 set kode ke simbol N, dimana *codeword* dar suatu set panjang m-1 bit dan mungkin memiliki beberapa prefix, dan *codeword* dari set lainnya memiliki panjang m-bit, dan prefiks yang berbeda. *Phase-in code* memberikan sebuah alphabet untuk N simbol, dimulai dengan sebuah tabel kode yang terdiri dari simbol-simbol. Tiap iterasi mengikuti langkah-langkah sebagai berikut:

a. Membuat table simbol, frekuensi sebagai langkah pertama untuk mengetahui iterasi yang akan dilakukan.
b. Membuat satu set simbol data n (atau hanya bilangan bulat 0 sampai n - 1) di mana $2^m \leq n < 2^m + 1$ untuk beberapa bilangan bulat m, $n = 2^m + p$ di mana $0 \leq p < 2^m - 1$ dan juga mendefinisikan $P = 2^m - p$.
c. Bilangan bulat P pertama 0 hingga $P - 1$ menerima *codeword* pendek dan sisa 2 p integer P hingga $n - 1$ ditetapkan dengan *codeword* panjang. Kode pendek adalah bilangan bulat 0 sampai $P - 1$, masing-masing dikodekan dalam $m - 1$ bit.
d. Membedakan antara kedua *codeword* dengan ketetapan nilai m-bit P, P +1, ..., $P + p - 1$, diikuti oleh bit ekstra, 0 atau 1.
e. Mengubah pesan teks menjadi kode di dalam *Phased-in code* [9].

2.5 Kompresi

Kompresi adalah proses mengubah data asli ke bentuk kode untuk menghemat penyimpanan dan waktu persyaratan untuk transmisi data [10]. Berdasarkan kemungkinan data yang sudah dikompresi dapat dikembalikan ke bentuk aslinya atau data sebelum dikompresi, maka teknik kompresi data dibagi menjadi dua jenis diantaranya kompresi *Lossless* dan kompresi *Lossy.*

a. Kompresi *Lossless* adalah kelas dari algoritma data kompresi yang memungkinkan data yang asli dapat disusun kembali dari data kompresi. Lossless data kompresi digunakan dalam berbagai aplikasi seperti format ZIP dan GZIP. *Lossless* juga sering digunakan sebagai komponen dalam teknologi kompresi data *lossy*. Kompresi *lossless* digunakan ketika sesuatu yang penting pada kondisi asli.
b. Kompresi *Lossy* adalah suatu metode untuk mengkompresi data dan mendekompresinya, data yang diperoleh mungkin berbeda dari yang aslinya tetapi cukup dekat perbedaaanya. *Lossy* kompresi ini paling sering digunakan untuk kompres data multimedia atau gambar diam). Sebaliknya, kompresi *lossless* diperlukan untuk data teks dan *file*, seperti catatan bank, artikel teks dan lainnya. Format kompresi *lossy* mengalami *generation loss* yaitu jika melakukan berulang kali kompresi dan dekompresi *file* akan menyebabkan kehilangan kualitas secara progresif. Hal ini berbeda dengan kompresi data *lossless*. Pengguna yang menerima *file* terkompresi secara *lossy* (misalnya untuk mengurangi waktu *download*) *file* yang diambil dapat sedikit berbeda dari yang asli di-*level bit* ketika tidak dapat dibedakan oleh mata dan telinga manusia untuk tujuan paling praktis [11].

2.6 Pesan Teks

Pesan teks atau Short Message Service (SMS) adalah salah satu komunikasi teks melalui telepon seluler. SMS merupakan salah satu media yang paling banyak digunakan saat ini. Selain murah, prosesnya juga berjalan cepat dan langsung sampai pada tujuan, tetapi selama ini SMS baru digunakan sebatas untuk mengirim dan menerima pesan antara sesama pemilik telepon seluler.Kemudahan penggunaan, variasi layanan, dan promosi yang cukup gencar dari operator seluler menjadikan SMS sebagai layanan yang sangat populer di masyarakat khususnya kalangan mahasiswa . Seiring dengan perkembangan teknologi dan kreativitas operator dan service provider, layanan SMS yang mulanya hanya untuk saling kirim pesan antara subscriber, kini berkembang dan lebih variatif, seperti layanan jajak pendapat, ringtone, SMS premium, mobile banking, ticketing dan layanan pendidikan [12].

2.7 Metode Exponential

Penghalusan *exponential* adalah teknik prediksi rata-rata bergerak dengan pembobotan dimana data diberi bobot

Journal Global Tecnology Computer
Vol 2, No. 2, April, 2023, Hal 42 - 53
ISSN: **2809-6118** (Online - Elektronik)
Website https://ejurnal.seminar-id.com/index.php/jogtc

oleh sebuah fungsi *exponential*. Penghalusan *exponential* merupakan prediksi rata-rata bergerak dengan pembobotan canggih, namun masih mudah digunakan.
Rumus penghalusan exponential dapat ditunjukkan sebagai berikut:

$F_I = F_{t-1} + \alpha (A_{t-1} - F_{t-1})$

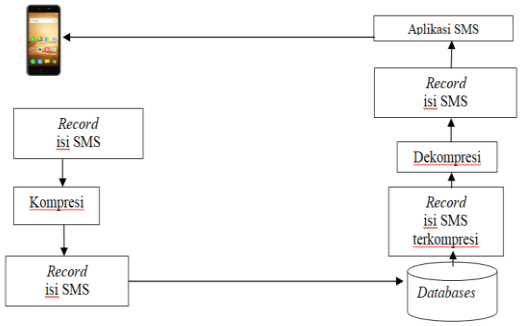
Keterangan :
F_I = Prediksi baru
F_{t-1} = Prediksi sebelumnya
A = Konstanta penghalus (pembobot) ($0 \leq \alpha \leq 1$)
A_{t-1} = Permintaan aktual periode lalu

Pendekatan penghalusan exponential mudah digunakan, dan telah berhasil diterapkan pada hampir setiap bisnis. Walaupun demikian, nilai yang tepat untuk konstanta penghalus, α dapat membuat diferensiasi antara prediksi yang akurat dan tidak akurat. Nilai α yang tinggi dipilih saat rata-rata cenderung berubah. Nilai α yang rendah digunakan saat rata-rata cenderung stabil. Tujuan pemilihan suatu nilai untuk konstanta penghalus adalah untuk mendapatkan prediksi yang akurat [13].

3. HASIL DAN PEMBAHASAN

3.1 Analisa

Dalam hal ini pengirim mengetik pesan sms yang akan dikompresi dengan memasukkan nomor tujuan dan selanjutnya pengirim mengkompres pesan tersebut atau pengirim juga bisa menekompresi pesan tersebut. Penerima pesan melihat isi pesan yang terkompresi dan mengetahui nomor tujuan si pengirim. Dengan teknik kompresi yang digunakan agar penyimpanan pesan teks tersebut semakin menghemat biaya dalam pengiriman pesan teks. Dengan menerapkan algoritma *tunstal code* dan *phased-in code*, yang merupakan kompresi data lossless yang memanipulasi bit data dalam sms untuk meminimalkan ukuran dengan cara membagi data menjadi dua keluaran, apabila persentase *byte* nol dari data masukan rendah, maka algoritma *tunstal code* dan *phased-in code*, akan menghasilkan pesan teks kompresi yang dapat mendekati nilai satu, bahkan bisa melebihi satu data terekspansi, sehingga dalam pengiriman pesan tidak banyak memakan kapasitas dan biaya yang di lebih besar.



Gambar 1. Prosedur Kompresi dan Dekompresi

Berdasarkan analisa, sebuah pesan SMS maksimal terdiri dari 140 bytes, dengan kata lain sebuah pesan bisa memuat 140 karakter 8-bit, 160 karakter 7-bit standar GSM 03.38 atau 70 karakter 16-bit standar Universal Character Set 2-byte (UCS-2) untuk bahasa Jepang, Bahasa Mandarin dan Bahasa Korea yang memakai Hanzi (Aksara Kanji / Hanja). SMS bisa dikirim melalui port tertentu, dengan memanfaatkan 56-64 bit dari total 140 bytes yang ada, jadi sebuah pesan hanya bisa memuat 133 karakter 8-bit, 152 karakter 7-bit (GSM 03.38) atau 66 karakter 16-bit (UCS-2). Adapula beberapa metode untuk mengirim pesan yang lebih dari 140 bytes, tetapi seorang pengguna harus membayar lebih dari sekali. SMS bisa pula untuk mengirim gambar, suara dan film. SMS bentuk ini disebut Multimedia Message Service (MMS). Pesan-pesan SMS dikirim dari sebuah telepon genggam ke pusat pesan atau *Short Message Service Center* (SMSC), di sini pesan disimpan dan dicoba dkirim selama beberapa kali. Setelah selama waktu yang telah ditentukan, biasanya 1 hari atau 2 hari, lalu pesan dihapus. Seorang pengguna bisa mendapatkan konfirmasi dari pusat pesan ini. Dalam penelitian ini, akan dibahas 2 proses utama yaitu proses kompresi dan dekompresi, dan peneliti akan mengkompresi sebuah pesan teks dengan algoritma *tunstall code* dan *phased-in code*. *Tunstall code* dan *phased-in code* merupakan salah satu algoritma yang termasuk di dalam metode *lossless* data. Berikut adalah contoh pesan teks yang akan dikompresi dan dekompresi.

SELAMAT MALAM

Gambar 2. Pesan Teks

Journal Global Tecnology Computer
Vol 2, No. 2, April, 2023, Hal 42 - 53
ISSN: **2809-6118** (Online - Elektronik)
Website https://ejurnal.seminar-id.com/index.php/jogtc

3.1.1 Analisa Proses Kompresi Pada Algoritma *Tunstall code*

Proses kompresi pesan teks dengan algortima *tunstall code* yang akan dilakukan adalah dengan mengunakan rumus sebagai berikut:

$N + k (N-1) \leq 2^n$.

Dimana:
n = Panjang *bit* yang diinginkan
N = Jumlah simbol
k = maks.

Langkah-langkah proses kompresi algoritma *tunstall code* dapat dilihat dari contoh dengan menggunakan hasil sebagai berikut:
Pesan Teks : SELAMAT MALAM = 104 *bit*

a. Membuat tabel simbol, frekuensi sebagai langkah pertama untuk mengetahui iterasi yang akan dilakukan.

Tabel 1. Tabel Simbol Dan Frekuensi

Simbol	Frekuensi
S	1
E	1
L	2
A	4
M	3
Sp	1
T	1

Jumlah karakter berjumlah 7 sehingga dibutuhkan untuk *tunstall code* adalah 7 *bit*.

b. Menghitung jumlah iterasi yang akan dilakukan dengan menggunakan rumus algoritma *tunstall code.*
$N + k (N-1) \leq 2^n$
$13+k(13-1) \leq 2^7$
$13 + 12\ k \leq 128$
$12k \leq 128-13= 115$
$k \leq 115/12= 9,58$
Berdasarkan perhitungan diatas, iterasi yang akan dilakukan maksimal 10 kali iterasi, Dalam penelitian ini iterasi yang akan dilakukan hanya sebanyak 7 kali iterasi, karena dalam 7 kali iterasi hasil sudah didapatkan.

c. Membuat tabel simbol dan probabilitas untuk melakukan iterasi, untuk mencari probabilitas adalah dengan membagi frekuensi kemunculan karakter dengan jumlah karakter.
S = 1/13 = 0,07 A = 4/13 = 0,30 Sp = 1/13 = 0,07
E = 1/13 = 0,07 M = 3/13 = 0,23
L = 2/13 = 0,15 T = 1/13 = 0,07

Tabel 2. Tabel Simbol Dan Probabilitas

Simbol	Probabilitas
S	0,07
E	0,07
L	0,15
A	0,30
M	0,23
T	0,07
Sp	0,07

d. Iterasi selanjutnya dengan mengalikan karakter dengan probabilitas tertinggi dan menggabungkannya dengan karakter lainnya.

Tabel 3. Iterasi Pertama

Simbol	Probabilitas	Simbol	Probabilitas
S	0,07	AS	0,021
E	0,07	AE	0,021
L	0,15	AL	0,045
M	0,23	AM	0,069
T	0,07	AT	0,021

Tabel 4. Lanjutan Iterasi Pertama

Simbol	Probabilitas
Sp	0,07
AA	0,09
Asp	0,021

Tabel 5. Iterasi Kedua

Simbol	Probabilitas	Simbol	Probabilitas
E	0,07	Asp	0,021
L	0,15	SS	0,0049
M	0,23	SE	0,0049
T	0,07	SL	0,0105
Sp	0,07	SA	0,021
AA	0,09	SM	0,0161
AS	0,021	ST	0,0049
AE	0,021	SSp	0,049
AL	0,045		
AM	0,069		
AT	0,021		

Tabel 6. Iterasi Ketiga

Simbol	Probabilitas	Simbol	Probabilitas
L	0,15	EE	0,0049
M	0,23	ES	0,0049
T	0,07	EL	0,0105
Sp	0,07	EA	0,021
AA	0,09	EM	0,0161
AS	0,021	ET	0,0049
AE	0,021	ESp	0,0049
AL	0,045		
AM	0,069		
AT	0,021		
Asp	0,021		
SS	0,0049		
SE	0,0049		
SL	0,0105		
SA	0,021		
SM	0,0161		
ST	0,0049		

Tabel 7. Iterasi Keempat

Simbol	Probabilitas	Simbol	Probabilitas
M	0,23	EM	0,0161
T	0,07	ET	0,0049
Sp	0,07	ESp	0,0049
AA	0,09	LL	0,0225
AS	0,021	LS	0,0105
AE	0,021	LE	0,0105
AL	0,045	LA	0,045
AM	0,069	LM	0,0345
AT	0,021	LT	0,0105
Asp	0,021	LSp	0,0105
SS	0,0049		
SE	0,0049		
SL	0,0105		
SA	0,021		
SM	0,0161		

ST	0,0049
SSp	0,0049
EE	0,0049
ES	0,0049
EL	0,0105
EA	0,021

Tabel 8. Iterasi Kelima

Simbol	Probabilitas	Simbol	Probabilitas	Simbol	Probabilitas
T	0,07	SSp	0,0049	MM	0,0529
Sp	0,07	EE	0,0049	MS	0,0161
AA	0,09	ES	0,0049	ME	0,0161
AS	0,021	EL	0,0105	ML	0,0345
AE	0,021	EA	0,021	MA	0,069
AL	0,045	EM	0,0161	MT	0,0161
AM	0,069	ET	0,0049	MSp	0,0161
AT	0,021	ESp	0,0049		
Asp	0,021	LL	0,0225		
SS	0,0049	LS	0,0105		
SE	0,0049	LE	0,0105		
SL	0,0105	LA	0,045		
SA	0,021	LM	0,0345		
SM	0,0161	LT	0,0105		
ST	0,0049	LSp	0,0105		

Tabel 9. Iterasi Keenam

Simbol	Probabilitas	Simbol	Probabilitas
Sp	0,07	LL	0,0225
AA	0,09	LS	0,0105
AS	0,021	LE	0,0105
AE	0,021	LA	0,045
AL	0,045	LM	0,0345
AM	0,069	LT	0,0105
AT	0,021	LSp	0,0105
Asp	0,021	MM	0,0529
SS	0,0049	MS	0,0161
SE	0,0049	ME	0,0161
SL	0,0105	ML	0,0345
SA	0,021	MA	0,069
SM	0,0161	MT	0,0161
ST	0,0049	MSp	0,0161
SSp	0,0049	TT	0,0049
EE	0,0049	TS	0,0049
ES	0,0049	TE	0,0049
EL	0,0105	TL	0,0105
EA	0,021	TA	0,021
EM	0,0161	TM	0,0161
ET	0,0049	TSp	0,0049
ESp	0,0049		

Tabel 10. Iterasi Ketujuh

Simbol	Probabilitas	Kode Tunstall	Simbol	Probabilitas	Kode Tunstall
AA	0,09	00000001	SM	0,0161	00001100
AS	0,021	00000010	ST	0,0049	00001101
AE	0,021	00000011	SSp	0,0049	00001110
AL	0,045	00000100	EE	0,0049	00100010
AM	0,069	00000101	ES	0,0049	00100011
AT	0,021	00000110	EL	0,0105	00100100
Asp	0,021	00000111	EA	0,021	00100101

SS	0,0049	00001000	EM	0,0161	00100110
SE	0,0049	00001001	ET	0,0049	00100111
SL	0,0105	00001011	ESp	0,0049	00101000
SA	0,021	00001011	LL	0,0225	00101010

Tabel 11. Iterasi Ketujuh Tabel Lanjutan

Simbol	Probabilitas	Kode Tunstall
LS	0,0105	00101011
LE	0,0105	00101100
LA	0,045	00101101
LM	0,0345	00101110
LT	0,0105	00101111
LSp	0,0105	00110000
MM	0,0529	00001111
MS	0,0161	00010000
ME	0,0161	00010001
ML	0,0345	00010010
MA	0,069	00010011
MT	0,0161	00010100
MSp	0,0161	00010101
TT	0,0049	00010110
TS	0,0049	00010111
TE	0,0049	00011000
TL	0,0105	00011001
TA	0,021	00011010
TM	0,0161	00011011
TSp	0,0049	00011100
SpSp	0,0049	00011101
SpS	0,0049	00011110
SpE	0.0049	00110001
SpL	0,0105	00110011
SpA	0,021	00110100
SpM	0,0161	00110101
SpT	0,0049	00110110

e. Kompresi dilakukan dengan mengubah Teks menjadi kode di dalam tabel *tunstall code* sehingga didapatkan hasil:
00001001 00101101 00010011 00011100 00000101 = 40 *bit*
Ukuran *file* awal sebelum dikompresi adalah 104 *bit*, sehingga rasio kompresinya adalah:
RC = (Ukuran File Asli)/(Ukuran File Terkompresi)
RC = 104/40 = 2,6
Jika dinyatakan dalam bentuk persen maka dituliskan dalam rumus sebagai berikut:
SS = (1 - (Ukuran File Terkompresi)/(Ukuran File Asli)) x 100%
SS = (1 - (40/104)) x 100%
SS = (1 - 0,38) x 100 %
SS = 0,62 x 100 %
SS = 62 %
Dari perhitungan di atas, dapat disimpulkan bahwa dengan menggunakan algoritma *tunstall code* karakter di atas dapat di kompresi sebanyak 62 %.

Tabel 12. Hasil Karakter File Teks Terkompresi

Biner	Dec	Char
00001001	9	
00101101	45	-
00010011	19	□
00011100	28	□
00000101	5	

3.1.2 Proses Dekompresi Tunstall Code

Proses dekompresi *tunstall code* diawali dengan meng-*input file tunstall code* yang akan didekompresi, kemudian kode-kode untuk setiap simbol dari *file tunstall code* akan dibaca. Kode-kode tersebut akan dicocokkan *bit*

per bit dengan kode pada simbol-simbol dari *file* asli. Berdasarkan tabel 3.19 iterasi ketujuh nilai yang sudah menjadi kode akan dibaca oleh sistem, kemudian menggantinya dengan nilai awal nya sebagai berikut:
Kode 00001001 akan diganti dengan nilai simbol 'SE',
Kode 00101101 akan digantikan dengan nilai simbol 'LA',
Kode 00010011 akan digantikan dengan nilai simbol 'MA',
Kode 00011100 akan digantikan dengan nilai simbol 'TSp',
Kode 00000101 akan digantikan dengan nilai simbol 'AM',
Hingga menghasilkan *file* asli yaitu *file* teks " SELAMAT MALAM"

3.1.3 Analisa Proses Kompresi Pada Algoritma *Phased-in code*

Satu bagian algoritma ini terdiri dari kode-kode pendek yang memiliki kekuatan dan dapat dikumpulkan untuk kompresi data dengan simbol yang memiliki probabilitas yang kemungkinan sama. Banyak kode prefix yang dijelaskan disini yang dikembangkan untuk kompresi tipe data tertentu. Kode ini biasanya berisi *codeword* dan cocok untuk kompresi data dimana setiap simbol itu memiliki kemungkinan yang berbeda. Langkah pertama di dalam algoritma ini adalah membuat n data simbol, dimana $n = 2^m$ (menunujukkan bahwa $m = \log_2 n$), dengan menetapkan kode m-bit sebagai *codeword* .Namun, jika $2^{m-1}<n<2^m$, maka $= \log_2 n$ bukan bilangan bulat.
Proses kompresi pesan teks dengan algortima *phased-in code* yang akan dilakukan adalah dengan mengunakan rumus sebagai berikut:

$$\mathbf{2^{m-1}<n<2^m}$$

Dimana:
n = Panjang *bit* yang diinginkan
m = $\log_2 n$

Langkah-langkah proses kompresi algoritma *phased-in code* dapat dilihat dari contoh dengan menggunakan hasil sebagai berikut:
Pesan Teks : SELAMAT MALAM = 104 *bit*

a. Membuat tabel simbol, frekuensi sebagai langkah pertama.

Tabel 13. Tabel Simbol Dan Frekuensi

Simbol	Frekuensi
S	1
E	1
L	2
A	4
M	3
Sp	1
T	1

b. Membuat satu set simbol data n (atau hanya bilangan bulat 0 sampai n - 1) di mana $2^m \leq n < 2^m + 1$ untuk beberapa bilangan bulat m, $n = 2^m + p$ di mana $0 \leq p < 2^m - 1$, dan juga mendefinisikan $P = 2^m - p$.
$2^m \leq n < 2^m + 1$
$= 2^{\log 2(7)} \leq 7 < 2^{\log 2(7)} + 1$
$= 2^{2,8} \leq 7 < 2^{3,8}$
$= 6{,}96 \leq 7 < 13{,}92$ atau dibulatkan menjadi $7 \leq 7 < 14$
Tabel berikut mengilustrasikan metode dan kinerja kompresinya. Tabel 14 mencantumkan nilai m, p, dan P. untuk n = 7, 8, 9, 10, 11, 12, dan 13.

Tabel 14. Parameter *phased-in code*

n	***m***	**$p = n - 2^m$**	**$P = 2^m - p$**
7	2	3	1
8	3	0	8
9	3	1	7
10	3	2	6
11	3	3	5
12	3	4	4
13	3	5	3

c. Membedakan antara kedua *codeword* dengan ketetapan nilai m-bit P, P +1, ..., P + p − 1, diikuti oleh bit ekstra, 0 atau 1.

Tabel 15 juga merupakan kunci untuk memperkirakan rasio kompresi dari kode-kode ini. Tabel tersebut menunjukkan kapan n mendekati pangkat, ada beberapa codeword pendek dan banyak codeword panjang, menunjukkan efisiensi yang rendah. Kapan n adalah pangkat 2, semua codeword panjang dan metode tidak menghasilkan kompresi apa pun (rasio kompresi 1). Kapan n sedikit lebih besar dari pangkat 2, ada banyak codeword pendek dan karenanya kompresi lebih baik.

Tabel 15. Tujuh *Phased-in code*

i	**n= 7**	**8**	**9**	**10**	**11**	**12**	**13**
0	00	000	000	000	000	000	000
1	010	001	001	0010	0010	0010	0010

Tabel 16. Tujuh *Phased-in code*

2	011	010	010	0011	0011	0011	0011
3	100	011	011	0100	0100	0100	0100
4	101	100	100	0101	0101	0101	0101
5	110	101	101	0110	0110	0110	0110
6	111	110	110	0111	0111	0111	0111
7		111	1110	1000	1000	1000	1000
8			1111	1001	1001	1001	1001
9				1010	1010	1010	1010
10					1011	1011	1011
11						1100	1100
12							1101

Ukuran total file n codeword bertahap untuk diberikan n adalah : P. × m + 2 p (m + 1) = (2 m + 1 - n) m + 2 (n - 2 m) (m + 1) dimana m = $\log_2$ n, sedangkan ukuran total ideal n codeword panjang tetap (menjaga dalam pikiran fakta $\log_2$ n, biasanya noninteger).

d. Membuat tabel untuk kode akhir *Phased-in code.* Dimungkinkan juga untuk membuat kode suffix bertahap. Dimulai dengan satu set kode panjang tetap dan mengubahnya menjadi dua set codeword dengan menghapus bit paling kiri dari beberapa codeword. Bit ini dihapus jika nilainya 0 dan jika penghapusannya tidak menimbulkan ambiguitas. Tabel 3.15 (di mana bit yang dihilangkan dicetak miring) mengilustrasikan contoh untuk 12 bilangan bulat nonnegatif pertama. Representasi berukuran tetap dari bilangan bulat ini membutuhkan lima bit, tetapi masing-masing dari delapan bilangan bulat 8 hingga 12 dapat diwakili oleh hanya empat bit karena kode 5-bit dapat mewakili 32 simbol dan ini hanya memiliki 12 simbol. Pemeriksaan sederhana memverifikasi bahwa, misalnya mengkodekan bilangan bulat 8 sebagai 1000 daripada 01000 tidak menimbulkan ambiguitas, karena tidak satupun dari 23 kode lainnya diakhiri dengan 1000. Sepertiga dari kata sandi dalam contoh ini sedikit lebih pendek, tetapi jika kita hanya mempertimbangkan 13 bilangan bulat dari 0 hingga 12, sekitar setengahnya akan membutuhkan empat bit, bukan lima. Efisiensi kode ini tergantung di mana n (jumlah simbol) terletak di interval [2_m, $2_{m+1} - 1$]. Suffix Phased-in code adalah kode akhir (jika c telah dipilih sebagai codeword, tidak ada kata kode lain yang diakhiri dengan c). Kode akhir dapat dianggap sebagai pelengkap kode awalan. Kode akhir *phased-in code* dapat dilihat seperti tabel dibawah ini:

Tabel 17. Kode akhir *phased-in code*

00000	00001	00010	00011
*0*1000	*0*1001	*0*1011	*0*1011
10000	10001	10010	0011

e. Kompresi dilakukan dengan mengubah Teks menjadi kode di dalam tabel *tunstall code* sehingga didapatkan hasil :
00000 00001 00010 00011 01000 01001 01011 01011 10000 10001 10010 0011 = 96 *bit*
Ukuran *file* awal sebelum dikompresi adalah 104 *bit*, sehingga rasio kompresinya adalah
RC = (Ukuran File Asli)/(Ukuran File Terkompresi)
RC = 104/96 = 1,08
Jika dinyatakan dalam bentuk persen maka dituliskan dalam rumus sebagai berikut :
SS = (1 - (Ukuran File Terkompresi)/(Ukuran File Asli)) x 100%
SS = (1 - (96/104)) x 100%
SS = (1 – 0,9) x 100 %
SS = 0,1 x 100 %
SS = 10 %
Dari perhitungan di atas, dapat disimpulkan bahwa dengan menggunakan algoritma *phased-in code* karakter di atas dapat di kompresi sebanyak 10 %.

3.1.4 Perbandingan Menggunakan Metode *Single Exponential*

Rumus yang digunakan pada *single exponential smoothing* adalah:

$$F_{t+1} = \alpha\, y_t + (1-\alpha)\, F_t$$

Dimana:
F_{t+1} = Nilai peramalan untuk periode t+1.
y_t = Data sebenarnya pada periode t.
F*t* = Nilai ramalan periode t.
α = konstanta penghalusan ($0 < \alpha < 1$).

Langkah-langkah proses perbandingan algoritma *tunstal code* dan *phased-in code* adalah sebagai berikut:

a. Proses Perhitungan *tunstall code*
Misalkan α = 0,2, maka dengan rumus $F_{t+1}=\alpha\, y_t+(1-\alpha)F_t$, diperoleh:
F_{5+1} $=0{,}2*y_t+(1-\alpha)*F_t$
$= 0{,}2*104 + (1-0{,}2)*40$
$= 20{,}8 + 32$
$= 52{,}8$

b. Proses Perhitungan *phased-in code*
Misalkan α = 0,2, maka dengan rumus $F_{t+1}=\alpha\, y_t+(1-\alpha)F_t$, diperoleh :
$F_{12+1}=0{,}2*y_t+(1-\alpha)*F_t$
$= 0{,}2*104 + (1-0{,}2)*96$
$= 20{,}8 + 76{,}8$
$= 97{,}6$

Tabel 18. Perbandingan metode *single exponential*

Alternatif	RC	SS	F_t	Rangking
Algoritma *Tunstall code*	2,6	62 %	52,8	1
Algoritma *Phased-in code*	1,08	10 %	97,6	2

4. KESIMPULAN

Berdasarkan hasil anasila yang telah djabarkan diatas pada bab-bab sebelumnya dan hasil pengamatan penulis berdasarkan rumusan masalah , maka didapatkan beberapa kesimpulan bahwa Prosedur dalam mengkompresi sebuah pesan teks dengan algoritma *tunstall code* dan *phased-in code* dilakukan dengan menginput pesan teks, lalu melakukan proses kompresi, setelah hasil dikompresi lalu dilakukan proses dekompresi hingga menghasilkan *output* atau hasil pesan teks semula sebelum dikompresi. Cara memampatkan atau mengecilkan ukuran pesan teks dengan algoritma *tunstal code* dengan menggambar tabel symbol frekuensi, menghitung jumlah iterasi, membuat tabel symbol dan probabilitas untuk melakukan iterasi, mengalikan karakter dengan probabilitas tertinggi dan menggabungkan karakter lainnya, mengubah teks menjadi kode lalu menghitung ukuran bit setelah di kompresi. Cara memampatkan ukuran pesan teks dengan algoritma *phased-in code* dengan memuat tabel symbol frekuensi, membuat satu set symbol data n, membedakan kedua *codeword*, membuat kode akhir, mengubah teks menjadi kode lalu menghitung ukuran bit setelah dikompresi. Berdasarkan penerapan metode *single exponential* dapat membuktikan bahwa suatu pesan teks yang memiliki ukuran yang cukup besar dapat dikompresi menjadi lebih kecil dari ukuran sebelumnya.

REFERENCES

[1] N. Ayuningtyas, “Implementasi kode Huffman dalam aplikasi kompresi teks pada layanan SMS,” *Jur. Tek. Inform. Inst. Teknol. Bandung. Bandung*, no. 13506048, 2007.
[2] M. Roslin and A. Neta, “Perbandingan Algoritma Kompresi Terhadap Objek Citra Menggunakan JAVA,” vol. 2013, no. November, pp. 224–230, 2013.
[3] S. T. Klein and D. Shapira, “On improving Tunstall codes,” *Inf. Process. Manag.*, vol. 47, no. 5, pp. 777–785, 2011, doi: 10.1016/j.ipm.2011.01.005.
[4] G. Qiu, K. M. Lam, H. Kiya, X. Y. Xue, C. C. J. Kuo, and M. S. Lew, “Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics): Preface,” *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 6298 LNCS, no. PART 2, 2010, doi: 10.1007/978-3-642-15696-0.
[5] P. B. Tarigan, “**済無**No Title No Title,” *J. Chem. Inf. Model.*, vol. 53, no. 9, pp. 1689–1699, 2013, doi: 10.1017/CBO9781107415324.004.
[6] P. D. Dr. Suarga, M.Sc.,M.Math., *Algoritma Dan Pemrograman*. Yogyakarta: ANDI, 2012.
[7] P. D. Dr. Suarga, M.Sc.,M.Math., *Algoritma Dan Pemrograman*. Yogyakarta: ANDI, 2012
[8] David Salomon & Giovani Motta, *Handbook Of Data Compression*. Springer, 2010.

[9] David Salomon & Giovani Motta, *Handbook Of Data Compression*. Springer, 2010.
[10] A. Putera and U. Siahaan, "JURNAL INFORMATIKA Vol. 10, No. 2, Jul 2016 IMPLEMENTASI TEKNIK KOMPRESI TEKS HUFFMAN," *J. Inform.*, vol. 10, no. 2, pp. 1251–1261, 2016..
[11] D. A. Yansyah and I. Pendahuluan, "Perbandingan Metode Punctured Elias Code," vol. 2, no. 6, pp. 33–36, 2015.