JURIKOM (Jurnal Riset Komputer), Vol. 8 No. 3, Juni 2021
e-ISSN 2715-7393 (Media Online), p-ISSN 2407-389X (Media Cetak)
DOI 10.30865/jurikom.v8i3.2834
Hal 64−70

Penerapan Algoritma Tunstall Code Untuk Mengkompresi File Teks

Juli Fitri, Surya Darma Nasution, Sumiaty Hutabarat

Teknik Informatika, Universitas Budi Darma, Medan, Indonesia
Email: fitrilaen@gmail.com
Submitted **14-02-2021**; Accepted **24-06-2021**; Published **29-06-2021**

Abstrak

Teks pada umumnya berisi rangkaian karakter dan dapat membentuk suatu kata, surat dan narasi. Misalnya, file 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 file teks, perlu dilakukan proses kompresi agar ukuran file teks tersebut menjadi lebih kecil. Salah satu teknik kompresi file yang dapat memampatkan file teks dengan baik adalah kompresi file menggunakan algoritma Tunstall code. 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. Dengan menggunakan algoritma Tunstall code, penulis berhasil mengkompresi suatu file teks dengan cukup baik, yaitu memampatkan ukuran file teks dari 80 bit menjadi 40 bit dengan hasil rasio kompresi pengurangan ukuran sebanyak 50%.

Kata Kunci: Kompresi; File Teks; Algoritma Tunstall Code

Abstract

Text generally contains a series of characters and can form a word, letter and narrative. For example, a text file that is large in size results in the sending process taking longer, and uses a large amount of memory space in its storage. So to save storage space and speed up the process of sending text files, it is necessary to do a compression process so that the text file size becomes smaller. One of the file compression techniques that can compress text files well is file compression using the Tunstall code algorithm. The Tunstall code algorithm work system is to determine the sequence of symbols for each codeword, take the highest probability and iterate as many as N (number of symbols) to be compressed. By using the Tunstall code algorithm, the authors managed to compress a text file quite well, namely compressing the text file size from 80 bits to 40 bits with a compression ratio of 50% reduction in size.

Keywords: Compression; Text Files; Tunstall Code Algorithm

1. PENDAHULUAN

Perpindahan data sangat mudah dilakukan pada saat ini, tetapi tempat penyimpanannya yang menjadi kendala sangat mendasar. Hal ini disebabkan oleh ukuran data yang ingin dipindahkan tidak sesuai dengan media penyimpanan yang tersedia. Ukuran file yang semakin besar menuntut para pemakai komputer untuk melakukan berbagai macam cara agar dapat menyimpan sejumlah file yang berukuran besar dalam media penyimpanan yang terbatas. Hal inilah yang menyebabkan file 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 sebelum menyimpan atau memindahkan data ke dalam media penyimpanan. Teks pada umumnya berisi rangkaian karakter dan dapat membentuk suatu kata, surat dan narasi. Misalnya, file teks 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 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 yang mengakibatkan kualitas data yang dihasilkan jauh lebih rendah dari kualitas data asli, sedangkan metode *lossless* tidak menghilangkan data selama proses kompresi terjadi, akibatnya kualitas data hasil kompresi tidak menurun. Salah satu algoritma kompresi yang dapat digunakan adalah algoritma *Tunstall Code* yang merupakan suatu algoritma yang membutuhkan masukan kata bersama dengan distribusi probabilitas untuk setiap masukan dan dikonstruksi sebagai pohon probabilitas yang akan diasosiasikan dengan huruf dari abjad masukan, algoritma ini juga dapat mengkompresi suatu file yang berukuran besar hingga memperkecil ukuran file tersebut.

Pada penelitian terdahulu Surya Darma Nasution (2016), Perancangan aplikasi kompresi file teks dengan menggunakan *Microsoft Visual Basic* 2008 memudahkan proses kompresi hanya dengan menginputkan file teks maka akan di proses sehingga menghasilkan file hasil kompresi beserta file headernya. Hasil file teks sebelum dan sesudah dikompresi setelah dibandingkan telah mencapai 50% rasio perbandingannya[1]. Muhammad Iqbal Dzulhaq , Aan Ahmad Andayani (2014), Berdasarkan uji coba yang telah dilakukan, dengan melakukan pengkompresian terhadap beberapa file teks dan gambar, algoritma *Lempel-Ziv-Welch* lebih bagus digunakan dalam pengkompresian file teks namun ada beberapa file yang tidak bisa dilakukan kompresi seperti docx dan xlsx, apabila dilakukan kompresi file tersebut bertambah besar, setelah dilakukan dekompresi ukuran menjadi normal kembali[2].

2. METODOLOGI PENELITIAN

2.1 Kompresi

JURIKOM (Jurnal Riset Komputer), Vol. 8 No. 3, Juni 2021
e-ISSN 2715-7393 (Media Online), p-ISSN 2407-389X (Media Cetak)
DOI 10.30865/jurikom.v8i3.2834
Hal 64−70

Proses kompresi merupakan proses mereduksi ukuran suatu data untuk menghasilkan representasi digital yang padat atau mampat (compact) namun tetap dapat mewakili kuantitas informasi yang terkandung pada data tersebut. Pada citra dan audio, kompresi ini mengarah pada minimalisasi jumlah bit rate untuk representasi digital [3].

2.2 File Teks

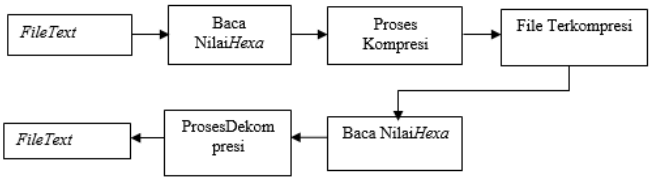
File teks merupakan file yang berisi informasi-informasi dalam bentuk teks. Data yang berasal dari dokumen pengolah kata, angka yang digunakan dalam perhitungan, nama dan alamat dalam basis data merupakan contoh masukan data teks yang terdiri dari karakter, angka dan tanda baca. Masukan dan keluaran data teks direpresentasikan[4].

2.3 Algoritma 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[5].
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:
1. s

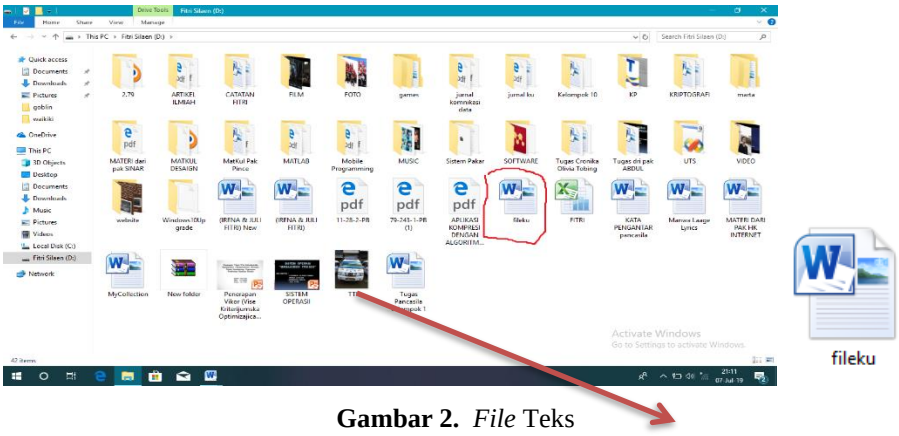
3. HASIL DAN PEMBAHASAN

Analisa terhadap sistem merupakan tahap yang sangat penting untuk mengetahui proses yang terjadi di dalam aplikasi yang akan dirancang, dalam hal ini peneliti menggunakan algoritma tunstall code untuk mengkompresi file teks.Pengompressan file berguna untuk mengurangi ruang penyimpanan serta mempercepat proses pengiriman suatu file. Dalam penelitian ini yang akan dibahas adalah bagaimana cara kerja algoritma Tunstall Code dalam mengkompresi file teks dengan menggunakan analisis kompresi. File teks yang akan digunakan dalam penelitian ini adalah file *.doc. File teks yang berekstensi *.doc mempunyai ukuran yang cukup besar, semakin banyak karakter yang ada dalam file, 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 file teks. Untuk mengetahui prosedur kompresi dan dekompresi suatu file teks dapat dilihat pada Gambar 1.



Gambar 1. Prosedur Kompresi Dekompresi *File Text*.

Dalam penelitian ini, akan dibahas 2 proses utama yaitu proses kompresi dan dekompresi, dan peneliti akan mengkompresi sebuah *file* teks dengan algoritma *Tunstall code*. *Tunstall code* merupakan salah satu algoritma yang termasuk di dalam metode *lossless* data. Berikut adalah contoh *file* teks yang akan dikompresi dan dekompresi.



Gambar 2. *File* Teks

JURIKOM (Jurnal Riset Komputer), Vol. 8 No. 3, Juni 2021
e-ISSN 2715-7393 (Media Online), p-ISSN 2407-389X (Media Cetak)
DOI 10.30865/jurikom.v8i3.2834
Hal 64−70

Proses kompresi *file* 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:

File teks : JULI FITRI = 80 *bit*

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

Tabel 1. Tabel Simbol Dan Frekuensi

Simbol	Frekuensi
J	1
U	1
L	1
I	3
Sp	1
F	1
T	1
R	1

Jumlah karakter berjumlah 8 sehingga dibutuhkan untuk *tunstall code* adalah 8 *bit*.

2. Menghitung jumlah iterasi yang akan dilakukan dengan menggunakan rumus algoritma *tunstall code.*

$N + k (N-1) \leq 2^n$
$10+k(10-1) \leq 2^8$
$10 + 9\ k \leq 256$
$9k \leq 256-10= 246$
$k \leq 246/9= 27,33$

Berdasarkan perhitungan diatas, iterasi yang akan dilakukan maksimal 27 kali iterasi, Dalam penelitian ini iterasi yang akan dilakukan hanya sebanyak 8 kali iterasi, karena dalam 8 kali iterasi hasil sudah didapatkan.

3. Membuat tabel simbol dan probabilitas untuk melakukan iterasi, untuk mencari probabilitas adalah dengan membagi frekuensi kemunculan karakter dengan jumlah karakter.

J = 1/10 = 0,1 I = 3/10 = 0,3 T = 1/10 = 0,1
U = 1/10 = 0,1 Sp = 1/10 = 0,1 R = 1/10 = 0,1
L = 1/10 = 0,1 F = 1/10 = 0,1

Tabel 2. Tabel Simbol Dan Probabilitas

Simbol	Probabilitas
J	0,1
U	0,1
L	0,1
I	0,3
Sp	0,1
F	0,1
T	0,1
R	0,1

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

Tabel 3. Iterasi Pertama

Simbol	Probabilitas	Simbol	Probabilitas
J	0.1	IJ	0,03
U	0.1	IU	0,03
L	0.1	IL	0,03
Sp	0.1	ISp	0,03
F	0.1	IF	0,03
T	0.1	IT	0,03
R	0.1	IR	0,03
II	0,09		

JURIKOM (Jurnal Riset Komputer), Vol. 8 No. 3, Juni 2021
e-ISSN 2715-7393 (Media Online), p-ISSN 2407-389X (Media Cetak)
DOI 10.30865/jurikom.v8i3.2834
Hal 64−70

Tabel 4. Iterasi Kedua

Simbol	Probabilitas	Simbol	Probabilitas
U	0.1	IF	0.03
L	0.1	IT	0.03
Sp	0.1	IR	0.03
F	0.1	JJ	0.01
T	0.1	JU	0.01
R	0.1	JL	0.01
II	0.09	JI	0.03
IJ	0.03	JSp	0.01
IU	0.03	JF	0.01
IL	0.03	JT	0.01
ISp	0.03	JR	0.01

Tabel 5. Iterasi Ketiga

Simbol	Probabilitas	Simbol	Probabilitas
L	0.1	JL	0.01
Sp	0.1	JI	0.03
F	0.1	JSp	0.01
T	0.1	JF	0.01
R	0.1	JT	0.01
II	0.09	JR	0.01
IJ	0.03	UU	0.01
IU	0.03	UJ	0.01
IL	0.03	UL	0.01
ISp	0.03	UI	0.03
IF	0.03	USp	0.01
IT	0.03	UF	0.01
IR	0.03	UT	0.01
JJ	0.01	UR	0.01
JU	0.01		

Tabel 6. Iterasi Keempat

Simbol	Probabilitas	Simbol	Probabilitas
Sp	0.1	JT	0.01
F	0.1	JR	0.01
T	0.1	UU	0.01
R	0.1	UJ	0.01
II	0.09	UL	0.01
IJ	0.03	UI	0.03
IU	0.03	USp	0.01
IL	0.03	UF	0.01
ISp	0.03	UT	0.01
IF	0.03	UR	0.01
IT	0.03	LL	0.01
IR	0.03	LJ	0.01
JJ	0.01	LU	0.01
JU	0.01	LI	0.03
JL	0.01	LSp	0.01
JI	0.03	LF	0.01
JSp	0.01	LT	0.01
JF	0.01	LR	0.01

Tabel 7. Iterasi Kelima

Simbol	Probabilitas	Simbol	Probabilitas	Simbol	Probabilitas
F	0.1	JSp	0.01	LI	0.03
T	0.1	JF	0.01	LSp	0.01
R	0.1	JT	0.01	LF	0.01
II	0.09	JR	0.01	LT	0.01
IJ	0.03	UU	0.01	LR	0.01
IU	0.03	UJ	0.01	SpSp	0.01

JURIKOM (Jurnal Riset Komputer), Vol. 8 No. 3, Juni 2021
e-ISSN 2715-7393 (Media Online), p-ISSN 2407-389X (Media Cetak)
DOI 10.30865/jurikom.v8i3.2834
Hal 64−70

IL	0.03	UL	0.01	SpJ	0.01
ISp	0.03	UI	0.03	SpU	0.01
IF	0.03	USp	0.01	SpL	0.01
IT	0.03	UF	0.01	SpI	0.03
IR	0.03	UT	0.01	SpF	0.01
JJ	0.01	UR	0.01	SpT	0.01
JU	0.01	LL	0.01	SpR	0.01
JL	0.01	LJ	0.01		
JI	0.03	LU	0.01		

Tabel 8. Iterasi Keenam

Simbol	Probabilitas	Simbol	Probabilitas	Simbol	Probabilitas
T	0.1	JR	0.01	SpSp	0.01
R	0.1	UU	0.01	SpJ	0.01
II	0.09	UJ	0.01	SpU	0.01
IJ	0.03	UL	0.01	SpL	0.01
IU	0.03	UI	0.03	SpI	0.03
IL	0.03	Usp	0.01	SpF	0.01
ISp	0.03	UF	0.01	SpT	0.01
IF	0.03	UT	0.01	SpR	0.01
IT	0.03	UR	0.01	FF	0.01
IR	0.03	LL	0.01	FJ	0.01
JJ	0.01	LJ	0.01	FU	0.01
JU	0.01	LU	0.01	FL	0.01
JL	0.01	LI	0.03	FI	0.03
JI	0.03	LSp	0.01	FSp	0.01
JSp	0.01	LF	0.01	FT	0.01
JF	0.01	LT	0.01	FR	0.01
JT	0.01	LR	0.01		

Tabel 9. Iterasi Ketujuh

Simbol	Probabilitas	Simbol	Probabilitas	Simbol	Probabilitas
R	0.1	UL	0.01	SpF	0.01
II	0.09	UI	0.03	SpT	0.01
IJ	0.03	USp	0.01	SpR	0.01
IU	0.03	UF	0.01	FF	0.01
IL	0.03	UT	0.01	FJ	0.01
ISp	0.03	UR	0.01	FU	0.01
IF	0.03	LL	0.01	FL	0.01
IT	0.03	LJ	0.01	FI	0.03
IR	0.03	LU	0.01	FSp	0.01
JJ	0.01	LI	0.03	FT	0.01
JU	0.01	LSp	0.01	FR	0.01
JL	0.01	LF	0.01	TT	0.01
JI	0.03	LT	0.01	TJ	0.01
JSp	0.01	LR	0.01	TU	0.01
JF	0.01	SpSp	0.01	TL	0.01
JT	0.01	SpJ	0.01	TI	0.03
JR	0.01	SpU	0.01	TSp	0.01
UU	0.01	SpL	0.01	TF	0.01
UJ	0.01	SpI	0.03	TR	0.01

Tabel 10. Iterasi Kedelapan

Simbol	Probabilitas	Kode Tunstall	Simbol	Probabilitas	Kode Tunstall
II	0.09	00000001	SpSp	0.01	00100010
IJ	0.03	00000010	SpJ	0.01	00100011
IU	0.03	00000011	SpU	0.01	00100100
IL	0.03	00000100	SpL	0.01	00100101
ISp	0.03	00000101	SpI	0.03	00100110
IF	0.03	00000110	SpF	0.01	00100111
IT	0.03	00000111	SpT	0.01	00101000

JURIKOM (Jurnal Riset Komputer), Vol. 8 No. 3, Juni 2021
e-ISSN 2715-7393 (Media Online), p-ISSN 2407-389X (Media Cetak)
DOI 10.30865/jurikom.v8i3.2834
Hal 64−70

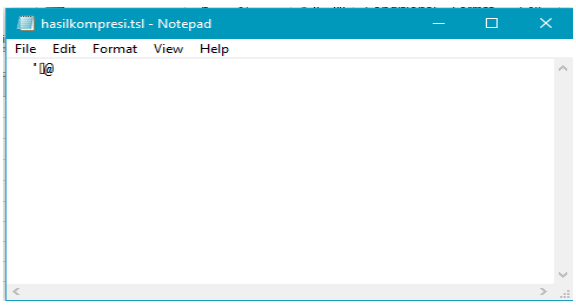
IR	0.03	00001000	SpR	0.01	00101010
JJ	0.01	00001001	FF	0.01	00101011
JU	0.01	00001010	FJ	0.01	00101100
JL	0.01	00001011	FU	0.01	00101101
JI	0.03	00001100	FL	0.01	00101110
JSp	0.01	00001101	FI	0.03	00101111
JF	0.01	00001110	FSp	0.01	00110000
JT	0.01	00001111	FT	0.01	00110001
JR	0.01	00010000	FR	0.01	00110011
UU	0.01	00010001	TT	0.01	00110100
UJ	0.01	00010010	TJ	0.01	00110101
UL	0.01	00010011	TU	0.01	00110110
UI	0.03	00010100	TL	0.01	00110111
USp	0.01	00010101	TI	0.03	00111000
UF	0.01	00010110	TSp	0.01	00111001
UT	0.01	00010111	TF	0.01	00111010
UR	0.01	00011000	TR	0.01	00111011
LL	0.01	00011001	RR	0.01	00111100
LJ	0.01	00011010	RJ	0.01	00111101
LU	0.01	00011011	RU	0.01	00111110
LI	0.03	00011100	RL	0.01	00111111
LSp	0.01	00011101	RI	0.01	01000000
LF	0.01	00011110	RSp	0.01	01000001
LT	0.01	00011111	RF	0.01	01000010
LR	0.01	00100001	RT	0.01	01000011

5. Kompresi dilakukan dengan mengubah Teks menjadi kode di dalam tabel *tunstall code* sehingga didapatkan hasil 00001010 00011100 00100111 00000111 01000000 = 40 *bit*
   Ukuran *file* awal sebelum dikompresi adalah 80 *bit*, sehingga rasio kompresinya adalah
   RC = (Ukuran File Asli)/(Ukuran File Terkompresi)
   RC = 80/40 = 2
   Jika dinyatakan dalam bentuk persen maka dituliskan dalam rumus sebagai berikut :
   SS = (1 - (Ukuran File Terkompresi)/(Ukuran File Asli)) x 100%
   SS = (1 - (40/80)) x 100%
   SS = (1 - 0,5) x 100 %
   SS = 0,5 x 100 %
   SS = 50%
   Dari perhitungan di atas, dapat disimpulkan bahwa dengan menggunakan algoritma *tunstall code* karakter di atas dapat di kompresi sebanyak 50 %.

Tabel 11. Hasil Karakter File Teks Terkompresi

Biner	Dec	Char
00001010	10	
00011100	28	
00100111	39	'
00000111	7	
01000000	64	@

Setelah nilai *biner* diketahui, maka mengubah nilai *biner* ke decimal dan mengubahnya lagi kedalam suatu karakter. Karakter hasil dari proses kompresi yang dihasilkan tersimpan dalam suatu *file* dengan ekstensi ".tsl", dan jika *file* tersebut dibuka dengan aplikasi *notepad*, maka akan tampil karakter seperti pada Gambar 3.



Gambar 3. Hasil Karakter Kompresi

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 3.10 iterasi kedelapan nilai yang sudah menjadi kode akan dibaca oleh sistem, kemudian menggantinya dengan nilai awal nya sebagai berikut:
Kode 00001010 akan diganti dengan nilai simbol 'JU',
Kode 00011100 akan digantikan dengan nilai simbol 'LI',
Kode 00100111 akan digantikan dengan nilai simbol 'SpF',
Kode 00000111 akan digantikan dengan nilai simbol 'IT',
Kode 01000000 akan digantikan dengan nilai simbol 'RI',
Hingga menghasilkan *file* asli yaitu *file* teks " JULI FITRI".

4. KESIMPULAN

Berdasarkan pembahasan dan evaluasi dapat disimpulkan prosedur dalam mengkompresi sebuah file teks dengan algoritma tunstall code dilakukan dengan menginput file teks, lalu melakukan proses kompresi, setelah hasil dikompresi lalu dilakukan proses dekompresi hingga menghasilkan output atau hasil file teks semula sebelum dikompresi. Berdasarkan penerapan algoritma tunstall code dapat membuktikan bahwa suatu file teks yang memiliki ukuran yang cukup besar dapat dikompresi menjadi lebih kecil dari ukuran sebelumnya.

REFERENCES

[1] F. Gram et al., "Analisis Perbandingan Kompresi dan Dekompresi Menggunakan Algoritma Shannon-Fano 2 Gram Dan Lempel Ziv Welch Pada Terjemahan Hadits Shahih Muslim," vol. 3, no. 3, pp. 5197–5204, 2016.
[2] M. S. B. S. G. Muhammad Iqbal Dzulhaq, Aan Ahmad Andayani , Dosen STMIK Bina Sarana global, Mahasiswa STMIK Bina Sarana Global, "Aplikasi Kompresi File Dengan Metode Lempel-Zif-Welchof," vol. 4, no. 1, pp. 1–4, 2014.
[3] R. N. Ibrahim, "PERBANDINGAN KOMPRESI FILE MENGGUNAKAN ALGORITMA RUN LENGTH DENGAN TWO LEVEL HOSHING," vol. 1, no. 2, pp. 90–104, 2007.
[4] David Salomon & Giovani Motta, Handbook Of Data Compression. Springer, 2010.
[5] I. M. Pu, Fundamentals Data Compression, Butterworth-Heinemann is an imprint of Elsevier, 2006.
[6] G. M. David Salomon and D. Bryant, Handbook of Data Compression Fifth Edition, 2010.
[7] R. S. Brar and B. singh, "A Survey on Different Compression Techniques and Bit Reduction Algorithm for Compression of Text/Lossless Data," International Journal of Advanced Research in Computer Science and Software Engineering, vol. 3, 2013.