ISSN : 2407 - 3911

IMPLEMENTASI ALGORITMA HUFFMAN DAN *RUN LENGTH ENCODING* PADA APLIKASI KOMPRESI BERBASIS *WEB*

Eka Prayoga, Kristien Margi Suryaningrum
Technology and Design Department, Informatics Engineering
Universitas Bunda Mulia
Jalan Lodan Raya No. 2, Ancol, Jakarta Utara
adrianuseka67@gmail.com, ksuryaningrum@bundamulia.ac.id

Abstrak

Meningkatnya penggunaan media *digital* dalam kehidupan sehari-hari secara tidak langsung turut meningkatkan kebutuhan dalam penyimpanan data, oleh karena itu dibutuhkan sebuah metode untuk menangani hal tersebut, salah satunya adalah dengan menerapkan kompresi data. Kompresi adalah teknik dalam memampatkan suatu data untuk menghemat media penyimpanan yang digunakan, selain itu, kompresi pun dapat dimanfaatkan untuk kebutuhan lain, seperti *backup data*, proses pengiriman data, serta keamanan data. Pemampatan atau kompresi pada umumnya diterapkan pada mesin komputer, karena setiap simbol yang ditampilkan memiliki bit-bit yang berbeda. Penulis menggunakan algoritma Huffman dan *Run Length Encoding* dalam proses pemampatan yang dilakukan, dimana masukkannya adalah *file* TXT. Tujuan penelitian ini adalah untuk mengetahui bagaimana implementasi dari gabungan antara kedua algoritma tersebut, selain itu, penelitian ini juga bertujuan untuk mengetahui bagaimana rasio perbandingan ukuran *file* antara *file* awal dan *file* yang terkompresi. Implementasi sistem yang dilakukan memanfaatkan aplikasi berbasis *web* untuk memudahkan pengguna dalam memanfaatkan fitur sistem yang ada, dimana dalam sistem ini memuat proses kompresi dan dekompresi. Tahapan kompresi digunakan untuk proses pemampatan, dan tahapan dekompresi untuk proses pengembalian *file* ke bentuk dan ukuran yang semula. Penelitian dilakukan dengan menggunakan 5 data uji, dan menunjukkan ukuran *file* hasil dekompres tidak seperti semula karena proses kompresi yang bersifat *lossy*.

Kata kunci :

Kompresi, TXT, Dekompresi, Huffman, Run Length Encoding

Abstract

Increasing the use of digital media in life indirectly also increases the need for data storage, therefore needed a method to handle it, one of them is by applying data compression. Compression is a technique which compress data to save used storage, in addition, any compression can be used for other needs, such as data backup, data transmission process, and data security. Compression or compression is generally applied to a computer machine, because every symbol displayed has different bits. Writer here used Huffman and Run Length Encoding algorithm in the compression process, where the input is TXT file. The purpose of this study is to find out how the implementation of the combination between the two algorithms, in addition, this study also aims to find out how the ratio of file sizes between the initial file and the compressed file. Implementation of the system made use of web-based applications to facilitate users in utilizing the features of existing systems, which in this system includes the compression and decompression process. The compression stages are used for the compression process, and the decompression stage for the process of returning the file to its original shape and size. The study was conducted using 5 test data, and showed the decompress file size is not as original because the compression process is categorized as lossy.

Keywords :

Compression, TXT, Decompression, Huffman, Run Length Encoding

I. PENDAHULUAN

Pertukaran informasi sekarang ini membutuhkan kecepatan. Salah satu cara untuk menangani hal ini adalah dengan teknik kompresi. Sekarang ini memang banyak aplikasi yang dapat melakukan kompresi secara efektif, namun dengan format yang sudah umum, tentu saja menimbulkan kerentanan dari segi keamanan, khususnya data yang memiliki sifat kerahasiaan yang tinggi. Selain itu, keharusan pengguna untuk melakukan registrasi serta pembelian nomor seri tentu saja menimbulkan pertimbangan dalam penggunaannya, meskipun pengguna dapat mencari nomor seri yang tidak resmi yang tentu saja dapat menimbulkan pelanggaran hak cipta. Secara umum, proses kompresi merupakan proses perubahan suatu data menjadi suatu kode, dimana dapat dikatakan efektif jika ukuran kode yang didapatkan lebih kecil disbanding ukuran kode yang aslinya.

Berawal dari permasalahan tersebut, penelitian ini dilakukan untuk mengimplementasikan algoritma kompresi yang ada pada sebuah aplikasi berbasis *web*, sehingga pengguna tidak harus mengunduh aplikasi tertentu, dan pengembang tidak harus khawatir terhadap adanya tindak pelanggaran hak kekayaan intelektual karena adanya *cracking* pada aplikasi yang ada. Selain itu, penelitian ini dilakukan untuk menguji implementasi algoritma Huffman dan *Run Length Encoding* pada aplikasi kompresi *string* berbasis *web*, membangun aplikasi kompresi yang dapat mengkompresi *file* serta dapat memperkecil besaran *file* sehingga media penyimpanan yang ada dapat digunakan untuk menyimpan *file* yang lebih banyak.

Penelitian ini dilakukan dengan beberapa batasan agar hasil yang didapatkan akurat, adapun batasan tersebut adalah penggunaan aplikasi berbasis *web* sebagai media implementasi algoritma.

II. TINJAUAN PUSTAKA

Pada subbab ini, akan dijelaskan mengenai konsep dari kompresi dan dekompresi, teori tentang algoritma yang digunakan, serta penjelasan singkat tentang aplikasi berbasis *web*, *website* dan bahasa pemrograman PHP.

Penelitian terdahulu yang sudah membahas mengenai topic ini adalah menurut Yana, Gusri dan Hondro (2013), pada penelitiannya mengenai Implementasi Algoritma Huffman dan LZ78 untuk Kompresi Data membahas mengenai besarnya ukuran data yang harus dikirim melampaui kecepatan transmisi yang dimiliki oleh perangkat keras yang ada, sehingga masih terdapat delay time yang relatife besar. Selain media penyimpanan seperti floppy disk, hard disk, dan compact disk mempunyai kapasitas yang terbatas. Oleh karena itu dengan menggunakan algoritma huffman dan LZ78 akan dilakukan kompresi data yang lebih kecil ukuran sizenya.

Kemudian penelitian lain, Menurut Asrianda (2012), membahas mengani kompresi file menggunakan algoritma huffman kanonik. Penelitian ini membahas mengenai Terdapat tiga fase dalam menggunakan algoritma Huffman untuk mengompres sebuah teks, pertama adalah fase pembentukan pohon Huffman, kedua fase encoding dan ketiga fase decoding. Prinsip yang digunakan oleh algoritma Huffman adalah karakter yang sering muncul di -encoding dengan rangkaian bit yang pendek dan karakter yang jarang muncul di-encoding dengan rangkaian bit yang lebih panjang. Teknik kompresi algoritma Huffman mampu memberikan penghematan pemakaian memori sampai 30%. Algoritma Huffman mempunyai kompleksitas O(n log n) untuk himpunan dengan n karakter.

Selain itu penelitian yang lain, oleh proses transformasi dalam bidang kompresi data dapat dianalogikan dengan proses enkripsi dalam bidang kriptografi. Algoritma Huffman sebagai salah satu algoritma kompresi data yang cukup tua memiliki prinsip-prinsip yang dapat diterapkan di bidang enkripsi data walaupun harus dilakukan beberapa adaptasi, terutama menyangkut implementasinya agar lebih praktis. Makalah ini akan membahas penerapan algoritma Huffman dalam dunia kriptogafi untuk akhirnya diambil sebuah kesimpulan apakah algoritma Huffman ini cukup layak dibahas sebagai algoritma kriptografi dilihat dari segi keamanan dan kepraktisannya.

II.1 Kompresi

Kompresi memiliki arti memperkecil ukuran atau memampatkan. Kompresi merupakan suatu proses pengkodean informasi menggunakan bit atau unit pembawa informasi yang lebih rendah daripada representasi data yang tidak terkodekan dengan suatu sistem *encoding* tertentu, sehingga data yang dikompresi lebih kecil sehingga dapat mempersingkat waktu dalam transmisi data tersebut. Umumnya, proses kompresi ini dilakukan atau diterapkan pada

sebuah mesin, khususnya komputer. Kompresi dibagi menjadi 2 (Krido Wicaksono & Pulung Nurtianto, 2015) yaitu:

1. *Lossy data-compression*
2. *Lossless data-compression*

Pada *lossy data*-compression, jika terjadi kesalahan pada proses kompresi, selama tidak mengubah pola dari data, maka kesalahan masih diperbolehkan, dan umumnya digunakan pada kompresi gambar dan suara. Namun, pada *lossless data-compression*, kesalahan tidak dapat ditoleransi, sehingga data sebelum dikompres harus sama persis dengan data hasil kompresi.

II.2 Dekompresi

Data yang telah dikompresi, harus dikembalikan seperti semula untuk kemudian dibaca kembali, proses pengembalian ini disebut sebagai proses dekompresi. Sama seperti kompresi, dekompresi dibagi menjadi dua, yaitu *lossy* dimana *file* yang dihasilkan, tidak sama persis seperti sebelum *file* dikompres, dan *lossless* dimana *file* yang dihasilkan akan sama persis seperti *file* sebelum dikompres.

II.3 Algoritma Huffman

Algoritma Huffman adalah algoritma yang dikembangkan David A. Huffman pada 1952. Algoritma Huffmn menggunakan prinsip pengkodean yang mirip dengan kode morse, yaitu tiap karakter (symbol) dikodekan hanya dengan rangkaian beberapa bit, dimana karakter yang sering muncul dikodekan dengan rangkaian bit yang pendek dan karakter yang jarang muncul dikodekan dengan rangkaian bit yang lebih panjang, karena prosesnya yang menggunakan kode ini, membuat algoritma Huffman sebagai algoritma keluarga dengan variable *codeword length* (Karisma Mahesa dan Karpen, 2017). Algoritma Huffman termasuk ke dalam kelas algoritma yang menggunakan metode static, yaitu metode yang selalu menggunakan peta kode yang sama. Metode ini membutuhkan dua fase, dimana fase pertama untuk menghitung probabilitas kemunculan tiap simbol dan menentukan peta kodenya, dan fase kedua untuk mengubah pesan menjadi kumpulan kode yang akan ditransmisikan.

Dalam mengkompresi dilakukan dengan menggunakan pemilihan algoritma Huffman. Hal ini dikarenakan metode Huffman merupakan salah satu teknik kompresi dengan cara melakukan pengkodean dalam bentuk bit untuk mewakili data karakter.

Cara kerja atau algoritma metode ini adalah sebagai berikut :

a. Menghitung banyaknya jenis karakter dan jumlah dari masing-masing karakter yang terdapat dalam sebuah file.
b. Menyusun setiap jenis karakter dengan urutan jenis karakter yang jumlahnya paling sedikit ke yang jumlahnya paling banyak.
c. Membuat pohon biner berdasarkan urutan karakter dari yang jumlahnya terkecil ke yang terbesar, dan memberi kode untuk tiap karakter.
d. Mengganti data yang ada dengan kode bit berdasarkan pohon biner.
e. Menyimpan jumlah bit untuk kode bit yang terbesar, jenis karakter yang diurutkan dari frekuensi keluarnya terbesar ke terkecil beserta data yang sudah berubah menjadi kode bit sebagai data hasil kompresi.

Contoh teknik kompresi dengan menggunakan metode Huffman pada file teks. Misalkan sebuah file teks yang isinya “AAAABBBCCCCCD”. File ini memiliki ukuran 13 byte atau satu karakter sama dengan 1 byte.

Berdasarkan pada cara kerja di atas, dapat dilakukan kompresi sebagai berikut :

a. Mencatat karakter yang ada dan jumlah tiap karakter. A = 4, B = 3, C = 12, D = 1
b. Mengurutkan karakter dari yang jumlahnya paling sedikit ke yang paling banyak yaitu : D, B, A, C
c. Membuat pohon biner berdasarkan urutan karakter yang memiliki frekuensi terkecil hingga yang paling besar.
d. Mengganti data yang ada dengan kode bit berdasarkan pohon biner yang dibuat.

Penggantian karakter menjadi kode biner, dilihat dari node yang paling atas atau disebut node akar :
A = 01, B = 001, C = 1, D = 000.

Selanjutnya berdasarkan pada kode biner masing-masing karakter ini, semua karakter dalam file dapat diganti menjadi :

0101010100100100111111000111111 1

Karena angka 0 dan angka 1 mewakili 1 bit, sehingga data bit di atas terdiri dari 32 bit atau 4 byte (1 byte = 8 bit)

e. Menyimpan kode bit dari karakter yang frekuensinya terbesar, jenis karakter yang

terdapat di dalam file dan data file teks yang sudah dikodekan.

Cara menyimpan data jenis karakter adalah dengan mengurutkan data jenis karakter dari yang frekuensinya paling banyak sampai ke yang paling sedikit, menjadi : [C,A,B,D] File teks di atas, setelah mengalami kompresi, memiliki ukuran sebesar 1 + 4 + 4 = 9 byte. Jumlah ini terdiri dari 1 byte kode karakter yang memiliki frekuensi terendah, 4 jenis karakter = 4 byte dan 4 byte data kode semua karakter.

II.4 Algoritma RLE

RLE (Run Length Encoding) adalah salah satu algoritma yang dapat digunakan untuk melakukan kompresi data sehingga ukuran data yang dihasilkan menjadi lebih rendah dari ukuran sebenarnya. Contoh yang dibahas kali ini adalah kompresi dan dekompresi dari sebuah kalimat atau *string* bentuk dari proses kompresi yang dilakukan dengan mengulangi *byte* yang sama secara terus-menerus (Dewi Indah Sari, 2016). Algoritma ini sangat berguna pada data yang memiliki banyak data dengan nilai yang sama secara berurutan seperti *file* ikon, gambar, garis dan animasi. Namun, algoritma ini tidak cocok diterapkan pada data normal karena akan mengakibatkan semakin bertambahnya ukuran data jika dibandingkan dengan data awalnya.

II.5 Aplikasi Berbasis *Web*

Aplikasi berbasis *web* merupakan aplikasi yang berjalan dengan memanfaatkan teknologi dari *internet* sehingga dapat dijalankan secara *portable*, aplikasi ini pada umumnya tidak membutuhkan banyak program tambahan yang harus dipasang di sisi *client* agar aplikasi berjalan. Aplikasi *web* juga merupakan suatu perangkat lunak yang dikodekan dalam bahasa pemrograman yang mendukung perangkat lunak berbasis web seperti HTML, *JavaScript*, CSS, Ruby, Python, PHP, Java dan bahasa pemrograman lainnya. Aplikasi berbasis *web* memiliki beberapa keuntungan, seperti:

1. Dapat digunakan dimanapun dan kapanpun tanpa mengharuskan pengguna melakukan instalasi apapun.
2. Tidak memerlukan lisensi pada saat penggunaanya.
3. Tidak terbatas pada sistem operasi apapun, serta tidak memerlukan spesifikasi yang tinggi, selama pengguna terhubung pada *internet*.

II.6 *Website*

Website adalah program yang dapat memuat film, gambar, suara, atau musik yang dapat ditampilkan di *internet*, sedangkan *web* adalah metode untuk menampilkan informasi di *internet*. Secara umum, *website* atau *world wide web* (www) dapat diartikan sebagai kumpulan halaman yang menampilkan informasi data teks, data gambar baik diam atau gerak, data animasi, suara, video, dan atau gabungan dari semuanya, baik yang bersifat statis maupun dinamis yang membentuk suatu rangkaian bangunan yang saling terkait dimana masing-masing dihubungkan dengan jaringan-jaringan halaman (*hyperlink*).

II.7 Bahasa Pemrograman PHP

PHP (*Hypertext Preprocessor*) adalah sebuah bahasa utama *script serverside* yang disisipkan pada HTML yang dijalankan di *server* dan juga bisa digunakan untuk membuat aplikasi *desktop*.

"PHP merupakan secara umum dikenal sebagai bahasa pemrograman *script-script* yang membuat dokumen HTML secara *on the fly* yang dieksekusi di *server web*, dokumen HTML yang dihasilkan dari suatu aplikasi bukan dokumen HTML yang dibuat dengan menggunakan teks atau *editor* HTML, dikenal juga sebagai bahasa pemrograman *server side"* (Betha Sidik, 2012).

II.8 Rasio Kompresi

Rasio kompresi merupakan perbandingan antara ukuran data yang telah berhasil dikompres dengan data sebelum dikompres dalam bentuk persentase. Secara sistematis, rasio kompresi ditunjukkan pada rumus [1].

$$R = 100\% - ((K_0 - K_1)/K_0) \times 100\%) \ldots [1]$$

Dimana R = Rasio Kompresi

K_0 = Ukuran data sebelum dikompres

K_1 = Ukuran data setelah dikompres

III. ANALISIS DAN PERANCANGAN

Adapun tujuan dari penelitian ini adalah memadatkan *string* yang ada dalam *file* txt sehingga ukuran *file* yang dikompres lebih kecil dan transmisi *file* tersebut dapat dilakukan dengan waktu yang lebih cepat.

Pembangunan sistem yang akan dirancang ini menggunakan model *waterfall*. Model pengembangan *waterfall* dianggap lebih sesuai karena dalam prosesnya pengerjaan sistem dilakukan secara bertahap, sehingga dalam pelaksanaannya, proses tidak saling tumpang tindih.

Berikut adalah tahapan-tahapan yang dilakukan pada perancangan sistem dengan model *waterfall*:

a) Desain Sistem (System Design)

Setelah kebutuhan sistem telah dirinci dan jelas, tahapan yang dilakukan adalah perancangan desain sistem agar sesuai dengan ketentuan yang telah dibuat, sehingga perancangan sistem pada tahapan selanjutnya dapat dilakukan dengan mudah.

b) Penulisan Kode Program

Bahasa pemrograman yang digunakan adalah bahasa pemrograman PHP, sedangkan aplikasi yang digunakan untuk menulis kode program adalah Notepad ++.

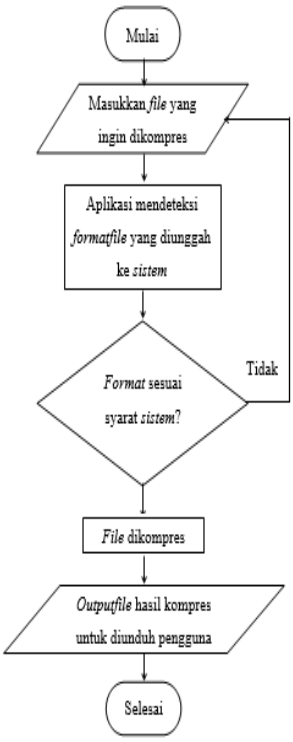
c) Pengujian (System Testing)

Pengujian dilakukan dengan metode black box.

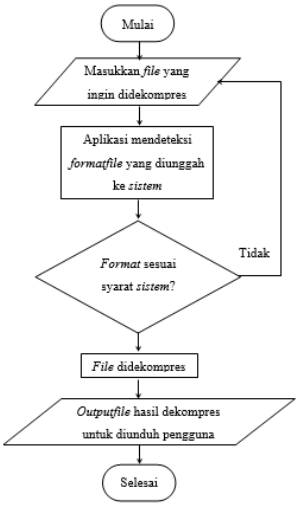
d) Pengoperasian dan Pemeliharaan (Operation and Maintenance)

Setelah sistem telah selesai dibangun secara keseluruhan, sistem akan digunakan, jika selama penggunaan terdapat masalah atau ada kekurangan, maka akan dilakukan penelitan serta evaluasi terhadap permasalahan serta pengembangan untuk menanggulangi masalah yang ada pada sistem tersebut.

Berikut adalah *flowchart* yang menjelaskan tentang proses kompresi dan dekompresi yang ada pada sistem. Proses kompresi pada aplikasi yang telah dibuat dijelaskan pada *flowchart* yang ada pada gambar 1 dan gambar 2.



Gambar 1. *Flowchart* Proses Kompresi pada Sistem



Gambar 2. *Flowchart* Proses Dekompresi pada Sistem

Flowchart diagram yang terlihat pada gambar 1 menjelaskan saat aplikasi dijalankan oleh hingga hasil

ditampilkan. Penjelasan secara detilnya adalah sebagai berikut:

a. Pengguna menjalankan aplikasi. Kegiatan ini ditandai dengan simbol "Mulai"
b. Pengguna mengunggah file asli yang akan dikompresi
c. Pengguna memasukkan nama file setelah dikompresi
d. Jika Pengguna memilih untuk kompresi, maka proses kompresi akan berjalan. Namun jika tidak maka proses akan selesai dilaksanakan

Sedangkan pada gambar 2 menjelaskan proses saat aplikasi menjalankan proses dekompresi, dijelaskan sebagai berikut:

a. Pengguna menjalankan aplikasi. Kegiatan ini ditandai dengan simbol "Mulai"
b. Pengguna mengunggah file asli yang akan di-dekompresi
c. Pengguna memasukkan nama file setelah di-dekompresi
d. Jika Pengguna memilih untuk mendekompresi file, maka proses akan berlanjut ke dekompresi, namun jika tidak maka proses akan berhenti.

Untuk penelitian ini, *database* diperlukan untuk menyimpan sementara *file* yang akan dikompres. Ditunjukkan pada Tabel 1 dan table 2.

Tabel 1 Tabel kompresi_file

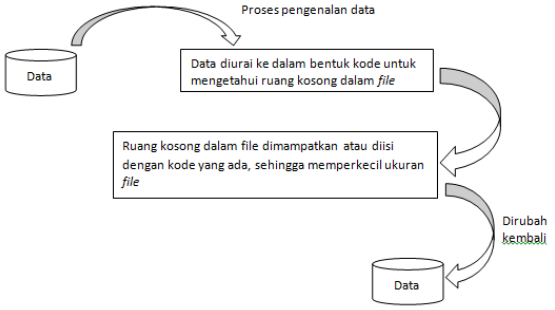
Field	Keterangan
kode_file	int, *primary key*
nama_file	varchar(10)
tipe_file	varchar(10)
ukuran_file_sblm	int
ukuran_file_ssdh	Int

Tabel 2. Tabel data_file

Field	Keterangan
Id	int, *primary key*, *auto increment*
kode_file	int
nama_file	varchar(10)
tipe_file	varchar(10)
ukuran_file	int

Field id merupakan *field* utama yang akan digunakan sebagai *unique id* antara data yang satu dengan data yang lainnya. *Field* nama_file digunakan untuk menyimpan indeks *array* nama *file* yang akan dikompres. *Field* tipe_file digunakan untuk menyimpan tipe *file* yang akan dikompres. *Field* ukuran_file_sblm digunakan untuk menyimpan ukuran *file* sebelum dikompres, sedangkan *field* ukuran_file_ssdh digunakan untuk menyimpan ukuran file sesudah dikompres.

ARSITEKTUR SISTEM

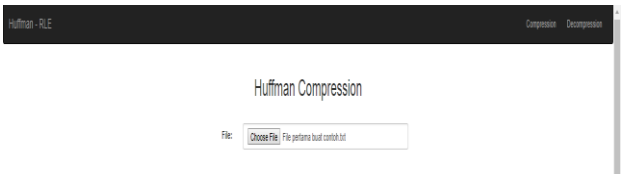


Gambar 3

Gambar 3 Arsitektur Sistem

Gambar 3.3 menunjukkan arsitektur sistem yang menggambarkan, data yang diunggah oleh pengguna akan dikenali, apakah data itu berupa *string*, gambar atau file lain. Setelah diketahui format datanya, data tersebut akan diurai kedalam bentuk kode, sehingga dalam rangkaian kode tersebut dapat diketaui ruang kosongnya atau kode yang dapat dihapus untuk proses kompresi. Setelah proses tersebut selesai dikerjakan, kode tersebut akan dirubah kembali ke dalam bentuk data seperti awalnya, namun memiliki ukuran yang lebih kecil.

PROSES *INPUT FILE* KOMPRESI



Gambar 4. Proses *Input*

Memilih *file* yang hendak dikompres dengan nama “file pertama buat contoh.txt”.

HASIL KOMPRESI

Gambar 5. Hasil Kompresi

Setelah *file* diunggah dan dikompresi, maka akan muncul persentase karakter dari *file* tersebut. Setelah kompresi selesai, akan muncul 2 *link* bagi *user* untuk mengunduh hasil kompresi dan *file frequency* yang berguna untuk proses dekompres. Jadi jika *user* menginginkan proses kompres dan dekompres, maka kedua *file* tersebut wajib untuk diunduh, sebaliknya jika hanya menginginkan hasil kompresinya saja, maka cukup unduh hasil kompresinya saja, ditunjukkan pada gambar 5

PROSES *INPUT FILE* DEKOMPRESI

Gambar 6 Input *file* untuk didekompres

Frequency file berguna sebagai kunci untuk proses dekompresi, maka dari itu ketika tadi *user* mengkompres *file* “file pertama buat contoh.txt” akan muncul 2 link untuk diunduh yaitu file “File_pertama_buat_contoh_convert.txt” dan “File_pertama_buat_contoh_frequency.txt”. Di bagian dekompresi ini, 2 file tersebut wajib untuk diunggah ulang jika *user* hendak mendekompresi.

PROSES DEKOMPRESI

Ascii Code	Char	Count
32		9
67	C	1
97	a	7
98	b	2
99	c	1
100	d	2
101	e	5
102	f	3
103	g	1
104	h	2
105	i	4
106	j	3
107	k	3
108	l	1
109	m	3
110	n	4
111	o	5
112	p	3
114	r	3
115	s	4
116	t	2
117	u	3
121	y	1

Your File has been Successfully Decompressed

Gambar 7. Hasil Dekompresi

Setelah *file convert* dan *frequency* diunggah ulang dan dikompres, maka akan kembali

menghasilkan *string* atau teks yang sama sesuai dengan file yang diunggah pada proses kompres.

Setelah proses kompresi dan dekompresi aplikasi yang dijelaskan melalui *flowchart* diatas, berikut adalah potongan *script* implementasi algoritma *Huffman* dan *Run Length Encoding* pada proses kompresi yang terdapat dalam aplikasi, yang ditunjukkan listing kode A dan listing kode B. ditunjukkan pada gambar 7.

```
function encode($str)
{
$encoded = '';
for ($i=0, $l=strlen($str), $cpt=0; $i<$l; $i++) {
if ($i+1<$l && $str[$i]==$str[$i+1] && $cpt<255) {
                                                $cpt++;
} else {
$encoded .= chr($cpt).$str[$i];
$cpt = 0;
}

}return $encoded;
}

while (!feof ($file))
{
$check;
$check = fgetc($file);
if($chrc=="\n" && feof ($file))
{
}
else
{
if($chrc=="0")
$bit[$i] = FALSE;
else
$bit[$i] = TRUE;
++$i;
$chrc = $check;
if($i==8)
{
for($j=0;$j<8;$j++)
$alpha = $alpha | ($bit[$j] << $j);
$ans_new = $ans_new.chr($alpha);
$i=0;
$alpha=chr(0);
} }
```

Listing Kode A Algoritma *Huffman* dalam proses Kompresi

Dan berikut adalah implementasi dengan menggunakan algoritma Huffman dan *Run Length Encoding* pada proses dekompresi yang ada pada aplikasi.

```
$data = array();
$total=0;
for($i=0;$i<256;$i++)
{
if($count_char[$i]!=0)
{
$data[$total] = new tree;
$data[$total]->freq = $count_char[$i];
$data[$total]->left = NULL;
$data[$total]->right = NULL;
$data[$total]->parent = NULL;
$data[$total]->alpha = chr($i);
$total++;
}
}
$result = huffman($data,$total);
$root = $result;
$bit = array();

or($j=0;$j<8;$j++)
$bit[$j] =FALSE;
$i=0;
if($chrc==chr(10) && $check>chr(48) &&
$check<chr(56))
{
$i=$check-chr(48);
$chrc = $prev;
}
else if($check==chr(10) || feof ($file))
{
$i=0;
$prev = $chrc;
}
else
{
$i=8;
}
for($j=0;$j<8;$j++)
$bit[$j] = ord($chrc) & (1<<$j);
for($j=0;$j<8;$j++)
if($bit[$j]>0)
$bit[$j]=1;
for($j=0;$j<$i;$j++)
{
if($root->left==NULL && $root->right==NULL)
{
fwrite($ready, $root->alpha);
```

```
$root = $result;
}
if($bit[$j]==0)
$root=$root->left;
else if($bit[$j]==1)
$root = $root->right;
}
$i=0;
$chrc = $check;
```

Listing Kode B Algoritma *Huffman* dalam proses Dekompresi

Implementasi algoritma pada aplikasi diuji dengan metode *black box*, dimana pengujian dilakukan sebanyak lima kali dengan lima bentuk *string* yang berbeda.

Pengujian dilakukan sebanyak 5 kali dengan 5 bentuk *string* yang berbeda

1. Awal: Contoh pertama file yang dijadikan subjek ujicoba proses kompres huffman
   Hasil Kompres:
   ‘µ±22¯ • ÁE–
   ò • ÆEè†Ð
   ————————————————‹Û}B[ëÃ
   • ÷ÇÛsæ¾ß-Ik]6
   Hasil Dekompres: Contoh pertama filiabg dijadikan subjek ujicoba proses kompres huffman
2. Awal: cccabbdddd
   Hasil Kompres: • Z 3
   Hasil Dekompres: cccabbdddd
3. Awal: abbcccddddeeeeeffffff
   Hasil Kompres: ‘Ù@Õÿ3
   Hasil Dekompres: abbcccddddeeeeeffffff
4. Awal: saat ini aku sedang mengetik file yang gatau isinya harus apaan tapi harus tetep diisi karena teks nya harus berisi dan gabisa kosong karena kalo kosong itu gatau mau kompres apaan
   Hasil Kompres:
   □흴밟截뤗時□律맵螞앆□뵔뿛랣□熓畫»
   鯏뷖甾嗩蕙笳捆毁bar鏺6 □□跉鋡諗□□□
   鎵춭뮊□劃
   Hasil Dekompres: saat ini aku sedang mengetik file yang gatau isinya harus apaan tapi harus tetep diisi karena teks nya harus berisi dan gabisa kosong karena kalo kosong itu gatau mau kompres apaan
5. Awal: hayah hayah payah amat lu ga tau mau isi apaan padahal kan tinggal ngetik aja gausa mikir aneh aneh cukup ngarang bebas aja disini mah
   Hasil Kompres: -´ßBûIhßh9À“/â£øÉs“î“Œ-
   xp_f‰
   ————————————————
   _âeÂe=‰ÙÇSÝå}—
   ÷k0}ÉÊ¥×yÝ|YÏLž • G5
   Hasil Dekompres: hayah hayah payah amat lu ga tau mau isi apaan padahal kan tinggal ngetik aja gausa mikir aneh aneh cukup ngarang bebas aja disini mah

Lalu, hasil perbandingan antara ukuran *file* sebelum dikompres, dan sesudah didekompres yang akan ditunjukkan pada tabel 3 dan tabel 4

Tabel 3 Hasil Pengujian Rasio Kompresi *File*

Nama File	Ukuran Awal	Ukuran Akhir	Rasio
File pertama buat contoh.txt	72 bytes	41 bytes	52,78%
File ke 2 yang dicoba.txt	10 bytes	5 bytes	50%
Encod.txt	21 bytes	9 bytes	28,57%
Tes4.txt	180 bytes	90 bytes	13,33%
Tes5.txt	134 bytes	65 bytes	5,22%

Tabel 4 Hasil Pengujian Dekompresi File

Nama File	Ukuran Awal	Ukuran Akhir
File pertama buat contoh.txt	41 bytes	71 bytes
File ke 2 yang dicoba.txt	5 bytes	11 bytes
Encod.txt	9 bytes	22 bytes

Tes4.txt	90 bytes	181 bytes
Tes5.txt	65 bytes	135 bytes

IV. KESIMPULAN DAN SARAN

Kesimpulan dari hasil penelitian ini adalah sebagai berikut:

1. Ukuran *file* hasil proses dekompresi menunjukkan bahwa *file* tidak kembali sepenuhnya seperti semula.
2. Ukuran *file* tidak menentukan rasio kompresi, rasio kompresi tergantung dari jumlah karakter yang ada pada *string* yang diproses.
3. Masih terdapat kesalahan pada hasil dekompresi, karena proses kompresi yang bersifat *lossy*.

Saran untuk peneliti selanjutnya adalah:

1. Implementasi dalam aplikasi masih dapat dikembangkan untuk proses kompresi *file* selain .txt.
2. Besaran *file* yang diunggah masih dapat diperbesar.

REFERENSI

Wicaksono, Krido, and Pulung Nurtantio Andono. 2016. Analisa Perbandingan Performa Kompresi Algoritma Huffman dan Algoritma LZW Terhadap Citra Medis MRI. Skripsi. Udinus

Kom, K.M., 2017. Rancang Bangun Aplikasi Kompresi Dan Dekompresi Pada Citra Digital Menggunakan Metode Huffman. *JURNAL PROCESSOR*, *12*(1), pp.997-1012.

Sari, Dwi Indah. "Perancangan Aplikasi Kompresi Citra dengan Metode Run Length Encoding untuk Keamanan File Citra Menggunakan Caesar Chipper", *Jurnal Ilmiah INFOTEK* 1.2 (2016).

Sidik, Betha (2012). Pemrograman Web dengan PHP, Informatika, Bandung

Rahandi, Aditya, Dian Rachmawati, and Sajadin Sembiring. "Analisis dan Implementasi Kompresi File Audio Dengan Menggunakan Algoritma Run Length Encoding (RLE)." *Alkhawarizmi* 1.1 (2012).

Hari Antoni Musril (2012), ISSN : 2086-4981, Volume : V, Studi Komparasi Metode *Aritmhatic Coding* & *Coding* dalam Algoritma *Entropy* untuk Kompresi Citra Digital.

Riyanto, M.Z., 2007. *Pengamanan Pesan Rahasia Menggunakan Algoritma Kriptografi Elgamal Atas Grup Pergandaan Zp**. Skripsi. Yogyakarta: Universitas Gajah Mada.