Journal of Computer System and Informatics (JoSYC)
Volume 1, No. 3, May 2020, Page 178-183
ISSN 2714-8912 (media online)
ISSN 2714-7150 (media cetak)

Penerapan Metode Lzw dan *Zero Compression* Untuk Mengkompresi *File Text*

Fachrurrazi

Program Studi Teknik Informatika, STMIK Budi Darma, Medan, Indonesia
Email: fachru.razi@gmail.com

Abstrak−*File text* sangat sering digunakan sehingga ukuran dari tiap *file* berbeda-beda sering terjadi permasalahan dalam media penampung *file*, semakin banyaknya *file text* dalam sebuah memori maka ukuran media penyimpanan juga semakin besar sehingga akan menjadi masalah. Kompresi adalah cara untuk meminimalkan ruang penyimpanan dalam merepresentasikan ukuran. Kompresi adalah teknik memadatkan data atau *file*, sehingga data atau *file* yang tadinya memiliki kapasitas data yang besar menjadi data yang lebih kecil, untuk menyimpan data atau *file* yang banyak pada memori yang memiliki kapasitas yang kecil. Algoritma lzw lebih baik dibandingkan dengan algoritma zero compression untuk mengkompresi file text.

Kata Kunci: Kompresi, Algoritma Lzw, Algoritma Zero Compression.

Abstract−Text files are very often used so that the size of each file is different, often there are problems in file storage media, the more text files in a memory, the larger the size of the storage media so that it will become a problem. Compression is a way to minimize storage space in representing size. Compression is a technique to compress data or files, so that data or files that have large data capacity become smaller data, to store large amounts of data or files in memory that has a small capacity. The lzw algorithm is better than the zero compression algorithm for compressing text files.

Keywords: Compression, LZW Algorithm, Zero Compression Algorithm.

1. PENDAHULUAN

File text merupakan file yang berisi karakter huruf yang sering dijumpai seperti file word, file pdf, file notepad. File text sangat sering digunakan sehingga ukuran dari tiap file berbeda-beda sering terjadi permasalahan dalam media penampung file, semakin banyaknya file text dalam sebuah memori maka ukuran media penyimpanan juga semakin besar sehingga akan menjadi masalah. Dalam mengatasi permasalahan tersebut dibutuhkan sebuah teknik dengan cara kompresi.

Masalah dalam penyimpanan adalah membutuhkan media penyimpan seperti memori, Oleh sebab itu, algoritma-algoritma kompresi yang berfungsi untuk memampatkan data hadir sebagai suatu solusi.

File teks seperti dokumen sangat sering digunakan dalam kehidupan sehari-hari untuk keperluan pribadi ataupun umum, dengan adanya file dokumen yang banyak maka akan membutuhkkan tempat penyimpanan yang besar pula sehingga menimbulkan permasalahan dalam media penyimpanan. Untuk mengatasi permasalahan ini maka diperlukan cara agar file dokumen yang besar dapat disimpan di memori yang kapasitasnya sedikit. Kompresi adalah solusi untuk mengatasi permasalahan pada file teks dengan begitu file teks yang telah dikompresi tersebut bisa lebih kecil dari ukuran sebelumnya.

Kompresi adalah teknik memadatkan data atau file, sehingga data atau file yang tadinya memiliki kapasitas data yang besar menjadi data yang lebih kecil, untuk menyimpan data atau file yang banyak pada memori yang memiliki kapasitas yang kecil. Kompresi data merupakan salah satu kajian di dalam ilmu komputer yang bertujuan untuk mengurangi ukuran file sebelum menyimpan atau memindahkan data tersebut ke dalam media penyimpanan. Kompresi data terdiri dari dua proses utama yaitu kompresi dan dekompresi atau pemulihan data kembali seperti aslinya. Jika suatu file dikompresi, maka file tersebut harus dapat dibaca kembali setelah file tersebut di dekompresi. Ada dua teknik yang dapat dilakukan dalam

melakukan kompresi data yaitu Lossless Compression dan Lossy Compression. Lossless Compression merupakan kompresi data dimana hasil dekompresi dari data yang terkompresi sama dengan data aslinya dan tidak ada informasi yang hilang. Sedangkan Lossy Compression adalah kompresi data di mana hasil dekompresi dari data yang terkompresi tidak sama dengan data aslinya karena ada informasi yang hilang, tetapi masih dapat ditolerir oleh persepsi mata. Kompresi data dicapai dengan mengurangi redudancy (kelebihan data) tapi ini juga membuat data kurang dapat diandalkan, lebih rentan terhadap kesalahan. Membuat data yang lebih handal, disisi lain dilakukan dengan menambahkan bit cek dan bit paritas, sebuah proses yang meningkatkan ukuran kode. Misalnya Zlib dan GZip yang hingga saat ini mengembangkan sebuah library berdasarkan algoritma Deflate, dimana penggunaannya dapat dipakai secara bebas dan dapat terintegrasi pada software yang dirancang.

Kompresi data berarti suatu teknik untuk memampatkan data agar diperoleh data dengan ukuran yang lebih kecil daripada ukuran aslinya sehingga lebih efisien dalam menyimpan serta mempersingkat waktu pertukaran data tersebut. Beberapa software kompresi yang banyak digunakan para pengguna komputer saat ini diantaranya adalah WinZip (menghasilkan format.zip) dan WinRAR (menghasilkan format.rar) dan lainnya[1].

Dengan adanya kompresi diharapkan dapat menghemat biaya serta waktu yang dikeluarkan guna menambah fasilitas media penyimpanan data pada komputer serta mempercepat proses transfer data.

Submitted: **05/05/2020**; Accepted: **18/05/2020**; Published: **26/05/2020**

Journal of Computer System and Informatics (JoSYC)
Volume 1, No. 3, May 2020, Page 178-183
ISSN 2714-8912 (media online)
ISSN 2714-7150 (media cetak)

Metode Kompresi dengan menggunakan Algoritma Lempel Ziv Welch (LZW) merupakan algoritma yang sering digunakan dalam proses kompressi data, pada beberapa penelitian terdahulu yang pernah dilakukan, di dalam jurnal yang berjudul Survey on LZWDictionary based Data Compression Technique, algoritma LZW disimpulkan merupakan algoritma yang memiliki rasio kompresi lebih baik dari algoritma kompresi dasar lainnya[2].

Menurut penelitan lainya, proses kompressi data dengan menggunakan algoritma LZW menghasilkan rasio kompresi sebesar 60-70% dari beberapa data baik itu data gambar maupun data text[3]. Sedangkan dalam penelitian lainya, Algoritma LZW menghasilkan waktu kompresi berkisar antara 55ms – 446 ms untuk data teks yang berisikan jumlah karakter antara 200 sampai 1000 karakter[4].

Zero Compression merupakan salah satu teknik kompresi yang diterapkan pada sekumpulan data sering terjadi perulangan data bernilai nol yang berurutan. Data yang digunakan dalam penelitian ini adalah data teks dengan alasan karena data teks lebih sederhana dalam pemrosesannya.

Alasan penulis menggunakan 2 metode dalam judul skripsi ini adalah, untuk mengetahui hasil rasio dari masing-masing metode dalam mengkompresi file teks, kemudian ingin membuat simulasi data kompresi dengan menggunakan metode algoritma LZW dan algoritma Zero Compression agar bisa mengetahui kelebihan dan kekurangan dari kedua algoritma tersebut sehingga dapat dipilih metode mana yang lebih efektif untuk mengkompresi file teks.

2. METODOLOGI PENELITIAN

2.1 Kompresi

Kompresi adalah proses mengubah data menjadi sekumpulan kode untuk menghemat tempat penyimpanan dengan atau tanpa mengurangi kualitas dari citra serta mempercepat waktu transmisi data. Pada penggunaannya, ada beberapa faktor yang sering menjadi pertimbangan dalam memilih suatu metode kompresi, yaitu kecepatan kompresi, sumber daya yang dibutuhkan (memori, kecepatan PC), ukuran *file* hasil kompresi serta kompleksitas algoritma, karena pada intinya tidak setiap metode kompresi sesuai untuk semua jenis *file* gambar. Tujuan dari kompresi terhadap citra digital ini adalah untuk mengurangi *redudansi* dari data-data yang terdapat dalam citra sehingga dapat disimpan atau ditransmisikan secara efisien. Masalah inilah yang menyebabkan munculnya kebutuhan akan kompresi citra tanpa harus mengurangi informasi yang tersimpan di dalam citra tersebut *lossless* [7].

2.2 File Teks

File teks merupakan *file* komputer yang tersusun atas rangkaian baris teks. Jenis-jenis *file* yang termasuk dalam kategori ini umumnya berisi rangkaian karakter tanpa informasi format *visual*. Jenis *file* yang termasuk kategori ini ada banyak. Beberapa diantaranya adalah sebagai berikut [10].

File-text merupakan *file* yang berisi informasi dalam bentuk *text*. Data yang berasal dari dokumen pengolah kata, angka yang digunakan dalam perhitungan, nama dan alamat dalam basis data merupakan contoh masukan data *text* yang terdiri dari karakter, angka dan tanda baca. Masukan dan keluaran data *text* direpresentasikan sebagai set karakter atau sistem kode yang dikenal oleh sistem komputer. Ada tiga macam set karakter yang umum digunakan untuk masukan dan keluaran pada komputer, yaitu ASCII, EBCDIC, dan *Unicode*. ASCII (*American Code for Information Interchange*) merupakan suatu standar internasional dalam kode huruf dan simbol seperti Hexdan Unicode, tetapi ASCII lebih bersifat universal. ASCII digunakan oleh komputer dan alat komunikasi lain untuk menunjukkan *text*. Kode ASCII memiliki komposisi bilangan biner sebanyak 8 *bit*, dimulai dari 00000000 hingga 11111111. Total kombinasi yang dihasilkan sebanyak 256, dimulai dari kode 0 hingga 255 dalam sistem bilangan desimal.

2.3 Metode *Zero Compression*

Pada metode *Zero Compression*, kompresi *file* audio dilakukan pada sampel audio yang bernilai nol (0) berurutan. Ada dua tahap utama kompresi dengan metode *Zero Compression* untuk data audio, yaitu *reading redudance* data dan *coding*. *Reading redudance* data adalah merepresentasikan frekuensi kemunculan setiap sampel audio kedalam bilangan eksak. *Coding* adalah menuliskan kode yang berisi nilai sampel dengan frekuensi kemunculannya [8]. Sebagai contoh diberikan sampel audio sebagai berikut:
0 0 0 0 0 2 4 5 2 0 0 0 0 7 8 9 1 adalah 17 byte.

Reading Redudance Data: 0 = 5, 2 = 1, 4 = 1, 5 = 1, 2 = 1, 0 = 4, 7 =1, 8 = 1, 9 = 1, 1 = 1 *Coding* sampel audio hasil kompresi adalah: 052452047891 adalah 12 *byte*.

2.4 Metode Lempel Zip Welch

Algoritma LZW menggunakan teknik adaptif dan berbasis “kamus”. Pendahuluan LZW adalah LZ77 dan LZ78 yang berkembang oleh Jacob Ziv dan Abraham Lempel pada 1997 dan 1978. Terry Welch mengembangkan teknik tersebut pada 1984. Algoritma ini melakukan kompresi dengan menggunakan *dictionary*, dimana *fragmen-fragmen* teks diganti dengan indeks yang diperoleh oleh sebuah “kamus”. Prinsip sejenis juga digunakan dalam

Submitted: **05/05/2020**; Accepted: **18/05/2020**; Published: **26/05/2020**

Journal of Computer System and Informatics (JoSYC)
Volume 1, No. 3, May 2020, Page 178-183
ISSN 2714-8912 (media online)
ISSN 2714-7150 (media cetak)

kode Braille, dimana kode-kode khusus digunakan untuk merepresentasikan kata-kata yang ada [11]. Pendekatan ini bersifat adaptif dan efektif karena banyak karakter dapat dikodekan dengan mengacu pada *string* yang telah muncul sebelumnya dalam teks. Prinsip kompresi tercapai jika referensi dalam bentuk *pointer* dapat disimpan dalam jumlah bit yang lebih sedikit dibandingkan string aslinya. Urutan langka algoritma kompresi LZW adalah sebagai berikut :

1. *Dictionary* diinisialisasikan dalam jumlah bit yang lebih sedikit dibandingkan string aslinya.
2. P adalah karakter pertama dalam *stream* karakter.
3. Q adalah karakter berikutnya dalam *stream* karakter.
4. Apakah string (P + Q) terdapat dalam *dictionary*?
   a. Jika "ya" maka P = P + Q (gabungan P dan Q menjadi *string* baru)
   b. Jika "tidak" maka:
   i. Output sebuah kode untuk menggantikan *string* P.
   ii. Tambahkan *string* (P + Q) ke dalam *dictionary* dan berikan nomor/kode berikutnya yang belum diganti dalam *dictionary* untuk *string* tersebut.
   iii. P = Q.
5. Apakah masih ada karakter berikutnya dalam *stream* karakter?
   a. Jika "ya" maka kembali ke langka 2.
   b. Jika "tidak" maka *output* kode yang menggantikan string P, lalu terminasi proses (*stop*).

3. HASIL DAN PEMBAHASAN

3.1 Analisa Masalah

Pembahasan yang akan dianalisa adalah teks sebelum dikompresi dan setelah dikompresi dengan menggunakan LZW dan *Zero compression* untuk dilakukannya analisa dari masing-masing aplikasi tersebut dengan tujuan untuk membandingkan ukuran teks atau rasio. Analisa perbandingan pada *aplikasi* LZW dan *Zero compression* dilakukan dengan tujuan agar didapat kelebihan dan kekurangan dari masing-masing *aplikasi* tersebut. Jenis teks yang akan dikompresi adalah *file text*.

Kompresi dilakukan untuk mengurangi kapasitas dari suatu *file* dimana ukuran dari sebuah *file* sangat mempengaruhi tempat penyimpanan, semakin besar ukuran *file* maka semakin besar pula tempat penyimpanan yang dibutuhkan oleh sebab itu diperlukan kompresi agar ukuran *file* bisa lebih kecil, sehingga media penyimpanan dapat menjadi lebih efesien.

3.2 Penerapan Metode LZW (Lempel Zip Welch)

Implementasi LZW begitu rumit, namun algortimanya begitu sederhana. Konsepnya adalah mengganti *string* pada suatu karakter dengan *code word* tunggal yang tersimpan pada kamus. Sebagian besar implementasi dari LZW menggunakan 12-bit *code word* untuk merepresentasikan 8-bit karakter masukan. 256 lokasi pertama digunakan untuk menyimpan karakter tunggal. Kombinasi baru terdapat pada deretan *input* akan ditambahkan ke dalam kamus dan akan disimpan pada lokasi 256 sampai 4.096.

Berikut ini karakter yang akan dikompresi adalah sebagai berikut :

Sebuah string akan di kompresikan dengan LZW **[fachrurrazi adalah mahasiswa budidarma]**.

Isi *dictionary* pada awal proses di *set* dengan 14 karakter dasar yang ada: yaitu {f,a,c,h,r,u,z,i, d,l, m,s w,b}. Tahapan kompresi ditunjukkan pada tabel 1.

Tabel 1. Tahapan Kompresi LZW

Langkah	Posisi,[kode] & karakter (P)	Gabungan posisi & Karakter (Q)	Dictionary (P + Q)	Kode	Output
1	inisialisasi		F	[1] F	
2			A	[2] A	
3			C	[3] C	
4			H	[4] H	
5			R	[5] R	
6			U	[6] U	
7			Z	[7] Z	
8			I	[8] I	
9			D	[9] D	
10			L	[10]L	
11			M	[11] M	
12			S	[12] S	
13			W	[13] W	

Submitted: **05/05/2020**; Accepted: **18/05/2020**; Published: **26/05/2020**

Journal of Computer System and Informatics (JoSYC)
Volume 1, No. 3, May 2020, Page 178-183
ISSN 2714-8912 (media online)
ISSN 2714-7150 (media cetak)

Langkah	Posisi,[kode] & karakter (P)	Gabungan posisi & Karakter (Q)	Dictionary (P + Q)	Kode	Output
14			B	[14] B	

Tabel 2. Tahapan Kompresi LZW

Langkah	Posisi,[kode] & karakter (P)	Gabungan posisi & Karakter (Q)	Dictionary (P + Q)	Kode	Output
1	[1] F	[A] 1 +2	F,A		[1]
2	[2] A	[C] 2+3	A,C		[2]
3	[3] C	[H] 3+4	C,H		[3]
4	[4] H	[R] 4+5	H,R		[4]
5	[5] R	[U] 5+6	R,U		[5]
6	[6] U	[Z] 6+7	U,Z		[6]
7	[7] Z	[I] 7+8	Z,I		[7]
8	[8] I	[D] 8+9	I,D		[8]
9	[9] D	[L] 9+10	D,L		[9]
10	[10] L	[M] 10+11	L,M		[10]
11	[11] M	[S] 11+12	M,S		[11]
12	[12] S	[W] 12+13	S,W		[12]
13	[13] W	[B] 13+ 14	W,B		[13]
14	[14] B	Habis			

Berdasarkan perhitungan kompresi di atas maka didapat hasil kompresi adalah sebagai berikut ini.

Tabel 3. Hasil Kompresi LZW

Dictionary	Output
F,A	[1]
A,C	[2]
C,H	[3]
H,R	[4]
R,U	[5]
U,Z	[6]
Z,I	[7]
I,D	[8]
D,L	[9]
L,M	[10]
M,S	[11]
S,W	[12]
W,B	[13]

Apabila dilakukan kompresi menggunakan algoritma LZw, maka hasilnya akan menjadi : [1], [2], [3], [4], [5], [6], [7], [8], [9], [10], [11], [12], [13]. Output [1] artinya di dictionary dikodekan menjadi F,A, output [2] artinya di dictionary dikodekan menjadi A,C begitu seterusnya sampai dengan out terakhir.

Total awal bit disimpan tanpa kompresi = Total *input* * *bit dictionary*
= 35 * 8
= 280 bit

Besar *file* setelah dikompresi = Total *output* * *bit dictionary*
= 13 * 8
= 104 bit

Hasil Rasio Kompresi :

$$\text{Rasio} = 100 - \frac{ukuran\ file\ terkompresi}{ukuran\ file\ asli}\ x\ 100\ \%$$

Rasio = 100 - (104 / 280) x 100 %
Rasio = 100 - (0,37) x 100%
Rasio =100 - 37 %
Rasio = 63 %

Berdasarkan perhitungan di atas didapat rasio 63% yang artinya memori berhasil menghemat memori sebesar 63 persen.

Submitted: **05/05/2020**; Accepted: **18/05/2020**; Published: **26/05/2020**

Journal of Computer System and Informatics (JoSYC)
Volume 1, No. 3, May 2020, Page 178-183
ISSN 2714-8912 (media online)
ISSN 2714-7150 (media cetak)

3.3 Penerapan Metode Zero Compression

Pada metode *Zero Compression*, kompresi *file* teks dilakukan pada sampel dengan cara mengurutkan frekuensi kemunculan data teks. Ada dua tahap utama kompresi dengan metode *Zero Compression* untuk data teks, yaitu *reading redudance* data dan *coding*. Kompresi data teks dengan menggunakan algoritma *zero compression* dapat dilihat berdasarkan perhitungan yang sesuai dengan algoritma tersebut. Secara algoritma teknik kompresi dapat diselesaikan dengan menggunakan perhitungan sebagai berikut:
Data Sampel = Fachrurrazi adalah Mahasiswa Budidarma

Tabel 4. Perhitungan Karakter Teks

Karakter	Jumlah	Hasil
F	1	F
A	10	A10
C	1	C
H	3	H3
R	4	R4
U	2	U2
Z	1	Z
I	3	I3
D	3	D3
L	1	L
M	2	M2
S	2	S2
W	1	W
B	1	B

Berdasarkan perhitungan kompresi di atas maka didapat hasil kompresi adalah sebagai berikut ini.

Tabel 5. Hasil Kompresi Perhitungan Karakter Text

Karakter	Output
F	F
A	A10
C	C
H	H3
R	R4
U	U2
Z	Z
I	I3
D	D3
L	L
M	M2
S	S2
W	W
B	B

Apabila dilakukan kompresi menggunakan algoritma *Zero Compression*, maka hasilnya akan menjadi : [F], [A10], [C], [H3], [R4], [U2], [Z], [I3], [D3], [L], [M2], [S2], [W] , [B]. Output [F] artinya karakter F diulang sebanyak satu kali, output [A10] artinya karakter A diulang sebanyak 10 kali, begitu seterusnya sampai dengan out terakhir.
Ukuran teks sebelum dikompresi = 35 x 8 bit = 280 Bit
Ukuran teks sesudah dikompresi = 23 x 8 bit = 184 Bit
Hasil Rasio Kompresi :

$$\text{Rasio} = 100 - \frac{ukuran\ file\ terkompresi}{ukuran\ file\ asli}\ x\ 100\ \%$$

Rasio = 100 – (280/184) x 100%
Rasio = 100 – (152) x 100%
Rasio = 100 – 152%
Rasio = 52%

Berdasarkan perhitungan di atas didapat rasio sebesar 52% yang artinya memori berhasil menghemat memori sebesar 52 persen.

Submitted: **05/05/2020**; Accepted: **18/05/2020**; Published: **26/05/2020**

4. KESIMPULAN

Beberapa kesimpulan yang dapat diambil dari penelitian yang telah dilakukan diatas diantaranya:

1. Algoritma lempel zip welch mempunyai dapat mengkompresi file text dengan lebih baik dibandingkan zero compression.
2. Algoritma zero compression dapat mengkompresi file text dengan sifat lossless.
3. Algoritma lempel zip welch dan algoritma zero compression adalah algoritma kompresi yang sifatnya lossless yang dapat diterapkan meggunakan Program Visual Basic Net 2008. Dan diharapkan dapat membantu pengguna dalam pengkompresian file teks.

REFERENCES

[1] U. Nurdin, Konteks Implementasi Berbasis Kurikulum, Jakarta: PT. Raja Grafindo Persada, 2002.
[2] T. S. dkk, *Pengolahan Citra Digital*, Yogyakarta: Andi, 2009.
[3] P. Danoedoro, *Pengantar Pengindraan Jauh Digital*, Yogyakarta: Andi, 2012.
[4] D. Putra, *Pengolahan Digital*, Yogyakarta: Andi, 2010.
[5] A. K. D. A. Susanto, *Teori dan Aplikasi Pengolahan Citra*, Yogyakarta: Andi, 2013.

Submitted: **05/05/2020**; Accepted: **18/05/2020**; Published: **26/05/2020**