Implementasi Algoritma Lempel-Ziv-Welch untuk Kompresi File Teks

Gusti Eka Yuliastuti[1], Danang Haryo Sulaksono[1], Ris Fani Kusuma[1], Sita Fara Yunanda[1]*

[1]Institut Teknologi Adhi Tama Surabaya

**Penulis korespondensi: gustiekay@itats.ac.id*

ABSTRACT

Data and information is very important for society currently. People need fast information through digital media. The longer it will be the more data and information stored. This has an impact on the lack of available storage space. The data and information need to be compressed into a smaller size to be more efficient in terms of storage. To perform the efficiency of data and information storage, the author performs compression with a case study of document files. File compression is done by applying the Lempel-Ziv-Welch Algorithm (LZW). The advantage of the LZW algorithm is that the compression results are good with a shorter time. The results of file compression with the LZW Algorithm will not affect the file, because the LZW Algorithm is included in the lossless type where the file results before and after compression will not differ significantly.

Article History

Received 09-08-2023
Revised 28-08-2023
Accepted 31-08-2023

Key words

Data
Dokumen
File Kompresi
Lempel-Ziv-Welch

ABSTRAK

Saat ini data dan informasi merupakan hal yang sangat penting bagi masyarakat. Masyarakat membutuhkan informasi yang cepat melalui media digital. Semakin lama akan semakin banyak data dan informasi yang disimpan. Hal tersebut berdampak pada kurangnya ruang penyimpanan yang tersedia. Data dan informasi tersebut perlu dipadatkan ke dalam ukuran yang lebih kecil agar lebih efisien dalam hal penyimpanan. Untuk melakukan efisiensi penyimpanan data dan informasi tersebut, penulis melakukan kompresi dengan studi kasus file dokumen. Kompresi file dilakukan dengan menerapkan Algoritma Lempel-Ziv-Welch (LZW). Kelebihan dari Algoritma LZW yakni hasil kompresi baik dengan waktu yang lebih singkat. Hasil kompresi file dengan Algoritma LZW tidak akan mempengaruhi file secara keseluruhan, karena Algoritma LZW termasuk ke dalam jenis lossless dimana hasil file sebelum dan sesudah kompresi tidak akan berbeda secara signifikan.

PENDAHULUAN

Data dan informasi merupakan hal yang sangat penting bagi masyarakat. Masyarakat membutuhkan informasi yang cepat melalui media digital. Semakin lama akan semakin banyak data dan informasi yang disimpan. Salah satu jenis data yang biasa disimpan adalah data teks. Penyimpanan data teks dalam bentuk digital menjadi salah satu pilihan yang terbaik, karena tidak membutuhkan tempat penyimpanan yang besar [1]. Selain itu, pemilik data teks dapat dengan mudah membuat, mengubah serta menghapus kapan saja dan dapat mengirim kemana saja. Karena hal itu data teks digital dapat dikatakan mempunyai fleksibilitas dan mobilitas yang sangat tinggi [2].

Kebutuhan masyarakat terhadap informasi yang cepat serta efisien sehingga data dan informasi tersebut disimpan dalam media digital. Hal tersebut berdampak pada kurangnya ruang penyimpanan yang tersedia. Semakin banyaknya informasi yang dibutuhkan maka secara otomatis diperlukan ruang penyimpanan yang cukup besar juga. Permasalahan yang dihadapi yakni kurang luasnya kapasitas ruang penyimpanan yang ada agar dapat menyimpan data dan informasi sebanyak–banyaknya. Data dan informasi tersebut perlu dipadatkan ke dalam ukuran yang lebih kecil agar lebih efisien dalam hal penyimpanan [3]. Untuk melakukan efisiensi penyimpanan data dan informasi tersebut, penulis melakukan kompresi dengan studi kasus data teks pada *file* dokumen. Jika ruang penyimpanan *file* dokumen tersebut lebih kecil maka akan memberi ruang kosong lebih pada media penyimpanan yang digunakan [4].

Terdapat berbagai macam algoritma yang dapat diterapkan untuk melakukan kompresi *file* dokumen, salah satunya yakni Algoritma Lempel-Ziv-Welch (LZW). Algoritma LZW merupakan jenis *lossless* dimana hasil *file* sebelum dan sesudah kompresi tidak akan berbeda secara signifikan [5]. Algoritma ini melakukan kompresi dengan menggunakan kamus, dimana fragmen–fragmen teks digantikan dengan indeks yang diperoleh dari sebuah “kamus” atau *dictionary*. Perdekatan ini

bersifat adaptif dan efektif karena banyak karakter dapat dikodekan tercapai jika referensi dalam bentuk *pointer* dapat disimpan dalam jumlah bit yang lebih sedikit dibandingkan string aslinya [6].

Pada penelitian sebelumnya yang dilakukan oleh Pratiwi pada tahun 2018, menjelaskan bahwa Algoritma LZW untuk kompresi file multimedia dapat dilakukan secara optimal. Kualitas file multimedia yang dihasilkan cukup baik, serta rasio pemampatan kompresi sebesar 84,84%. Sedangkan pada penelitian yang dilakukan oleh Deepa pada tahun 2019, menjelaskan bahwa Algoritma LZW merupakan salah satu algortima yang kuat serta banyak digunakan. Algoritma LZW menghasilkan rasio kompresi yang lebih baik dengan masuknya fitur pendekatan secara praktis. Berdasarkan latar belakang tersebut, penulis akan menerapkan Algoritma LZW untuk proses kompresi *file* dokumen pada penelitian ini.

TINJAUAN PUSTAKA

Kompresi

Kompresi data merupakan cabang ilmu komputer yang bersumber dari teori informasi. Teori informasi sendiri dipelopori oleh Claude Shannon sekitar tahun 1940-an. Teori ini mendefenisikan jumlah informasi di dalam pesan sebagai jumlah minimum bit yang dibutuhkan yang dikenal sebagai *entropy*. Teori ini juga memfokuskan pada berbagai metode tentang informasi termasuk tentang redundancy (informasi tak berguna) pada suatu pesan dimana semakin banyak redundancy maka semakin besar pula ukuran pesan, upaya mengurangi redundancy inilah yang akhirnya melahirkan subjek ilmu tentang kompresi data [7].

Menurut David Salomon, data kompresi adalah proses pengkonversian *input data stream* (sumber *stream* atau data mentah asli) ke bentuk *stream* lain (*output stream* atau *stream* yang sudah terkompresi) yang memiliki ukuran lebih kecil. Stream merupakan sebuah *file* ataupun *buffer* dalam memori. Sedangkan menurut Ida Mengyi Pu, kompresi data adalah konteks dari ilmu komputer, ilmu (dan seni) yang menyajikan informasi ke dalam bentuk yang *compact*. Pada umumnya data kompresi terdiri dari pengambilan *stream* simbol dan mengubahnya ke dalam bentuk kode. Jika kompresi efektif, maka hasil kode *stream* akan lebih kecil daripada simbol asli [8].

Pada dasarnya kompresi bertujuan untuk mengubah ukuran data menjadi lebih kecil dari ukuran semula, sehingga dapat menghemat ruang penyimpanan dan waktu untuk transmisi data [9]. Beberapa contoh penerapan kompresi data dapat dilihat seperti pada gambar-gambar yang diperoleh dari internet yang biasanya dalam format JPEG (*Joint Photographic Experts Group*) atau GIF (*GraphicalInterchange Format*), modem yang sebagian besar menggunakan kompresi, HDTV (*High Defenition Television*) akan dikompresi menggunakan MPEG-2 (*MovingPicture Experts Group 2*), serta beberapa teks sistem yang secara otomatis dikompresi ketika *file* tersebut disimpan [5].

Lossless Compression

Lossless Compression merupakan suatu metode kompresi data dengan tidak adanya "informasi" data yang hilang atau berkurang jumlahnya selama proses kompresi. Sehingga setelah proses dekompresi jumlah bit (*byte*) data atau informasi dalam keseluruhan data hasil sama persis dengan data aslinya. Kompresi jenis ini tidak selalu dapat mengurangi ukuran data secara berarti, karena tidak semua data mengandung informasi yang tidak perlu. Selain itu, data yang telah dikompresi tidak dapat dikompresi lagi. *Lossless compression* biasanya digunakan untuk mengurangi ukuran data-data yang penting, data-data yang isinya tidak boleh berubah sedikitpun. Misalnya data- data yang berbentuk dokumen, teks, *spreadsheet*, kode sumber (*source code*), serta data-data program [7].

METODE

Algoritma Lempel-Ziv-Welch (LZW)

Algoritma Lempel-Ziv-Welch (LZW) merupakan algoritma *lossless compression* yang ditemukan oleh Abraham Lempel, Jacob Ziv dan Terry Welch. Algoritma ini diciptakan oleh Welch tahun 1984 sebagai implementasi dan pengembangan dari algoritma LZ78 yang telah diciptakan sebelumnya oleh Lempel dan Ziv pada tahun 1978. Algoritma LZW dirancang untuk cepat dalam implementasi tetapi biasanya tidak optimal karena hanya melakukan analisis pada data [10].

Algoritma LZW melakukan kompresi dengan menggunakan *dictionary*, dimana fragmen-fragmen teks digantikan dengan indeks yang diperoleh dari sebuah kamus. Prinsip sejenis juga digunakan dalam kode *Braille*, dimana kode-kode khusus digunakan untuk mempresentasikan kata-kata yang ada. Pendekatan ini bersifat adaptif dan efektif karena banyak karakter bisa dikodekan dengan mengacu pada *string* yang telah muncul sebelumnya dalam teks [11].

Prinsip kompesi tercapai jika referensi dalam bentuk pointer dapat disimpan dalam jumlah bit yang lebih dibandingkan string aslinya. Sebagai contoh, string "ABBABABAC" akan dikompresi dengan LZW. Isi *dictionary* pada awal proses diatur dengan tiga karakter dasar yang ada: "A", "B", "C". Tahapan proses kompresi ditunjukkan pada Tabel 1.

Tabel 1. Proses Kompresi

Step	Posisi	Karakter	*Dictionary*	*Output*
1	1	A	[4] A B	A
2	2	B	[5] B B	B
3	3	B	[6] B A	B
4	4	A	[7] A B A	A B
5	6	C	[8] A B A C	A B A
6	9	C	---	C

Kolom posisi menyatakan posisi sekarang dari *stream* karakter dan kolom karakter menyatakan karakter yang terdapat pada posisi tersebut. Kolom *dictionary* menyatakan *string* baru yang sudah ditambahkan ke dalam *dictionary* dan nomor indeks untuk *string* tersebut ditulis dalam kurung siku. Kolom *output* menyatakan kode *output* yang dihasilkan oleh langkah kompresi. Sedangkan proses dekompresi pada Algoritma LZW dilakukan dengan prinsip yang sama seperti proses kompresi. Tahapan dekompresi ditunjukkan pada Tabel 2.

Tabel 2. Proses Dekompresi

Step	Posisi	*Output*	*Dictionary*
1	1	A	---
2	2	B	[4] A B
3	2	B	[5] B B
4	4	A B	[6] B A
5	7	A B A	[7] A B A
6	3	C	[8] A B A C

Prinsip umum kerja Algoritma LZW yakni memeriksa setiap karakter yang muncul kemudian menggabungkan dengan karakter selanjutnya menjadi sebuah *string* [12]. Jika *string* baru tersebut tidak berada dalam *dictionary* atau belum dimasukkan ke dalam indeks, maka *string* baru tersebut akan dimasukkan ke dalam indeks pada *dictionary*. Adapun alur dari Algoritma LZW sebagai berikut:

1. Kamus diinisialisasi dengan semua karakter dasar yang ada: {'A'..'Z','a'..'z','0'..'9'}.
2. P ← karakter pertama dalam *stream* karakter.
3. C ← karakter berikutnya dalam *stream* karakter.

4. Apakah *string* (P + C) terdapat dalam *dictionary*?
   - Jika ya, maka P ← P + C (gabungkan P dan C menjadi *string* baru).
   - Jika tidak, maka:
     i. *Output* sebuah kode untuk menggantikan *string* P.
     ii. Tambahkan *string* (P + C) ke dalam *dictionary* dan berikan nomor/kode berikutnya yang belum digunakan dalam *dictionary* untuk *string* tersebut.
     iii. P ← C.
5. Apakah masih ada karakter berikutnya dalam *stream* karakter?
   i. Jika ya, maka kembali ke langkah 2.
   ii. Jika tidak, maka *output* kode yang menggantikan *string* P, lalu terminasi proses (*stop*).

Compression Rate (CR)

Compression Ratio (CR) dilakukan dengan menghitung ukuran data *file* dokumen setelah dikompresi dan membaginya dengan ukuran *file* asli, setelah itu hasilnya dikalikan dengan 100%. Formula untuk menghitung RC seperti ditunjukkan pada Persamaan 1.

$$CR = \frac{ukuran\ file\ kompresi}{ukuran\ file\ asli} \times 100\% \quad \text{..... (1)}$$

Mean Square Error (MSE)

Mean Square Error (MSE) merupakan salah satu parameter untuk mengevaluasi kualitas dari *file* dokumen yang telah dikompresi. Jika nilai dari MSE kecil, maka kualitas dari *file* dokumen yang dikompresi akan semakin baik. Fungsi untuk menghitung MSE seperti ditunjukkan pada Persamaan 2.

$$MSE = \frac{1}{MN} \sum_{x=1}^{M} \sum_{y=1}^{N} [f(x,y) - f'(x,y)]^2 \quad \text{..... (2)}$$

HASIL DAN PEMBAHASAN

Pengujian pada kompresi *file* dokumen menggunakan Algoritma LZW ini meliputi pengujian sistem dengan data *testing*, dimana data yang digunakan sebagai *sample* sebanyak 20 data berupa data *file* berekstensi .pdf dan .doc. Pengujian dilakukan dengan cara melakukan proses kompresi menggunakan Algoritma LZW dan dilanjutkan dengan melakukan proses dekompresi menggunakan Algoritma LZW juga. Performansi dari hasil rekonstruksi diukur berdasarkan nilai *Compression Ratio* (CR) dan *Mean Square Error* (MSE).

Nilai CR bisa didapatkan dengan cara menghitung perbandingan nilai antara ukuran *file* sebelum dan sesudah kompresi. Adapun hasil pengujian nilai CR seperti ditunjukkan pada Tabel 3.

Tabel 3. Hasil Pengujian *Compression Ratio*

Nama *File*	*File* Hasil	*File* Asli	CR (%)
Data uji (1).docx	26	26	100
Data uji (2).pdf	333	331.8	99.6396396
Data uji (3).docx	17	17	100
Data uji (4).pdf	130	129	99.2307692
Data uji (5).pdf	130	130	100
Data uji (6).pdf	809	805	99.5055624
Data uji (7).pdf	6811	6804	99.8972251
Data uji (8).pdf	4754	4749	99.8948254
Data uji (9).pdf	16	16	100
Data uji (10).pdf	1554	1550	99.7425997

Nama *File*	***File* Hasil**	***File* Asli**	**CR (%)**
Data uji (11).pdf	370	369	99.7297297
Data uji (12).pdf	357	356	99.719888
Data uji (13).doc	24	24	100
Data uji (14).pdf	86	85	98.8372093
Data uji (15).docx	15	15	100
Data uji (16).pdf	365	362	99.1780822
Data uji (17).doc	267	261	97.752809
Data uji (18).docx	73	72	98.630137
Data uji (19).docx	14	14	100
Data uji (20).docx	14	14	100
Rata - rata	808.25	806.49	99.5879238

Berdasarkan Tabel 3, Nilai CR merupakan rasio kompresi (*Compression Ratio*), *File* Hasil adalah ukuran *file* sesudah dikompresi dan *File* Asli adalah ukuran *file* sebelum dikompresi. Rasio kompresi terbaik terjadi pada *file* yang memiliki nilai CR sebesar 100% (bobot karakter hampir sebesar atau sama besar dengan ukuran *file* yang sesungguhnya). Rasio kompresi terburuk terjadi pada Data Uji (17) yaitu sebesar 97,752809%, dimana hal tersebut juga terjadi pada beberapa *file* yang memiliki variasi karakter besar. Hasil pengujian rasio kompresi rata – rata yang didapatkan yakni sebesar 99,5879238%. Rasio kompresi berfungsi untuk menguji kesamaan ukuran setelah *file* hasil kompresi dikembalikan seperti semula atau bisa disebut dengan dekompresi.

Mean Square Error (MSE) merupakan nilai performansi yang harus dicari untuk mengetahui rekonstruksi *file* hasil kompresi. Selain itu, MSE juga digunakan untuk mengukur kesalahan proses kompresi secara kesuluruhan. MSE didapatkan dengan cara menghitung nilai rata – rata selisih kuadrat antara nilai yang dikompresikan dengan *file* aslinya. Hasil pengujian nilai MSE seperti ditunjukkan pada Tabel 4.

Tabel 4. Hasil Pengujian *Mean Square Error*

Nama *File*	***File* Hasil**	***File* Asli**	***Error Absolut***
Data uji (1).docx	26	26	0
Data uji (2).pdf	333	331.8	0.72
Data uji (3).docx	17	17	0
Data uji (4).pdf	130	129	0.5
Data uji (5).pdf	130	130	0
Data uji (6).pdf	809	805	8
Data uji (7).pdf	6811	6804	24.5
Data uji (8).pdf	4754	4749	12.5
Data uji (9).pdf	16	16	0
Data uji (10).pdf	1554	1550	8
Data uji (11).pdf	370	369	0.5
Data uji (12).pdf	357	356	0.5
Data uji (13).doc	24	24	0
Data uji (14).pdf	86	85	0.5
Data uji (15).docx	15	15	0
Data uji (16).pdf	365	362	4.5
Data uji (17).doc	267	261	18
Data uji (18).docx	73	72	0.5
Data uji (19).docx	14	14	0
Data uji (20).docx	14	14	0
Rata – rata MSE			**3.936**

Berdasarkan Tabel 4, Didapatkan nilai rata – rata Mean Square Error (MSE) yang cukup rendah yaitu 3,936. Hal ini menunjukkan bahwa Algoritma LZW cukup baik dalam hal rendahnya nilai error. Mean Square Error (MSE) berfungsi untuk menghitung rekonstruksi file kompresi dan mengukur kesalahan proses kompresi secara keseluruhan. Dengan nilai MSE terburuk terjadi pada Data uji (7) yaitu 24,5.

KESIMPULAN

Kinerja metode pada Algoritma LZW dapat diketahui dari nilai rasio kompresi yang diperoleh. Rasio kompresi terbaik terjadi pada *file* yang memiliki nilai CR sebesar 100% dimana artinya bobot karakter hampir sebesar atau sama besar dengan ukuran *file* yang sesungguhnya. Rasio kompresi terburuk terjadi pada Data Uji (17), dimana memiliki nilai CR sebesar 97,752809%, dimana didalam hal ini juga terjadi pada beberapa file yang memiliki variasi karakter besar. Dengan rasio kompresi rata – rata yang didapatkan adalah 99,5879238%. Rasio kompresi berfungsi untuk menguji kesamaan setelah file hasil kompresi dikembalikan seperti semula atau dekompresi.

Performa dari Algoritma LZW dapat diketahui dari nilai *Mean Square Error* (MSE). Dalam perhitungan MSE didapatkan nilai yang cukup rendah yakni sebesar 3,936. Hal ini menunjukkan bahwa Algoritma LZW memiliki performansi yang sangat baik.

DAFTAR PUSTAKA

[1] L. V Simanjuntak, "Perbandingan Algoritma Elias Delta Code dengan Levenstein Untuk Kompresi File Teks," *J. Comput. Syst. Informatics*, vol. 1, no. 3, pp. 184–190, 2020, [Online]. Available: https://ejurnal.seminar-id.com/index.php/josyc/article/view/168.

[2] D. H. Sulaksono, "Multiple Encryption dengan Menggunakan Metode Vigenere Chiper dan Blowfish," *SCAN*, vol. XI, no. 1, pp. 25–30, 2016.

[3] Y. Darnita, K. Khairunnisyah, and H. Mubarak, "Kompresi Data Teks dengan Menggunakan Algoritma Sequitur," *Sistemasi*, vol. 8, no. 1, p. 104, 2019, doi: 10.32520/stmsi.v8i1.429.

[4] A. R. Taqwa and D. H. Sulaksono, "Implementasi Kriptografi Dengan Metode Elliptic Curve Cryptography (ECC) untuk Aplikasi Chatting dalam Cloud Computing Berbasis Android," *KERNEL J. Ris. Inov. Bid. Inform. dan Pendidik. Inform.*, vol. 1, no. 1, pp. 42–48, 2020, doi: 10.31284/j.kernel.2020.v1i1.929.

[5] A. Fahrizon and M. A. Rony, "Implementasi Algoritma Enkripsi RSA dan Kompresi LZW pada Database Nasabah Di PT. Central Capital Futures," *Skanika*, vol. 1, no. 1, pp. 346–351, 2018, [Online]. Available: https://jom.fti.budiluhur.ac.id/index.php/SKANIKA/article/view/203.

[6] R. Mubarak, "Implementasi Sistem Keamanan Data Berbasis Kriptografi Rivest Code 6, Vigenere Chipper dan Kompresi Data LZW," *Esit*, vol. XV, no. 11, pp. 54–62, 2020, [Online]. Available: http://jurnal-eresha.ac.id/index.php/esit/article/viewFile/185/143.

[7] W. E. Pangesti, G. Widagdo, D. Riana, and S. Hadianti, "Implementasi Kompresi Citra Digital dengan Membandingkan Metode Lossy dan Lossless Compression Menggunakan Matlab," *J. Khatulistiwa Inform.*, vol. 8, no. 1, pp. 53–58, 2020, doi: 10.31294/jki.v8i1.7759.

[8] I. G. S. Astawa, "Penerapan Metode LZW dalam Kompresi File Chat Multimedia pada Sistem E-Marketplace Sarana Upakara Bali," *J. Ilmu Komput.*, vol. 11, no. 2, p. 68, 2018, doi: 10.24843/jik.2018.v11.i02.p02.

[9] A. Suharso, J. Zaelani, and D. Juardi, "Kompresi File Menggunakan Algoritma Lempel Ziv Welch (LZW)," *Komputasi J. Ilm. Ilmu Komput. dan Mat.*, vol. 17, no. 2, pp. 372–380, 2020, doi: 10.33751/komputasi.v17i2.2147.

[10] A. N. Alim, H. Yuana, and F. Febrinita, "Aplikasi Kompresi Citra dengan Menggunakan Algoritma Lempel Ziv Welch (LZW)," *J. Mhs. Tek. Inform.*, vol. 6, no. 2, pp. 684–695, 2022.

[11] A. A. Pirnando and R. Febryansyah, "Analisis Perancangan Desain Komputasi Paralel dengan Algoritma Lempel Ziv Welch (LZW)," *Teknol. Pint.*, vol. 2, no. 10, pp. 1–16, 2022.

[12] D. Oktaviani and I. M. Suartana, "Implementasi Kompresi Data dengan Modifikasi Algoritma Lempel-Ziv-Welch (LZW) untuk File Dokumen," *J. Informatics Comput. Sci.*, vol. 1, no. 03, pp. 128–137, 2020, doi: 10.26740/jinacs.v1n03.p128-137.