Implementasi Algoritma Shannon-Fano untuk Kompresi File Teks

Muhammad Khoiri[1], Gusti Eka Yuliastuti[2], Citra Nurina Prabiantissa[3], Muhammad Farid Firmansyah[4]
[1,2,3,4] Program Studi Teknik Informatika, Fakultas Teknik Elektro dan Teknologi Informasi,
Institut Teknologi Adhi Tama Surabaya
Email: [1]khoirimuh091@gmail.com , [2]gustiekay@itats.ac.id, [3] citranurina@itats.ac.id ,
[4]fansyahhh@gmail.com

Abstract. *The compression of text files is an important topic in processing data and transmitting information efficiently. The researchers analyzed and implemented the Shannon-Fano algorithm to compress text files. This research aims to reduce the size of a text file by utilizing the Shannon-Fano coding principle. The research method includes steps to implement the Shannon-Fano algorithm, which involves making a table of the frequency of occurrence of symbols in a text file and building a Shannon-Fano tree. A code-splitting technique was applied to build an optimal Shannon-Fano code based on the frequency of occurrence of symbols in a text file. This research method also analyzed the performance of the Shannon-Fano algorithm based on factors such as compression and decompression speed, memory usage, and time complexity. Various data tests have been used to test the performance of the algorithm, including text files of different sizes. The results of the performance analysis indicated that the Shannon-Fano algorithm had a high compression speed and efficient memory consumption. The Shannon-Fano algorithm was effective in compressing text files with significant results. Text file sizes could be efficiently reduced without sacrificing decompression accuracy. After implementing this algorithm, the average text file size could decrease by 24.31%, with the compression ratio reaching 75.68%, or 1:32. It indicated that the Shannon-Fano algorithm could provide a good compression level for text files.*
Keywords: *compression, data processing, shannon-fano algorithm, text file*

Abstrak. Kompresi *file* teks merupakan topik yang penting dalam hal pengolahan data dan pengiriman informasi secara efisien. Peneliti melakukan analisis dan implementasi dari algoritma Shannon-Fano untuk melakukan kompresi *file* teks. Tujuan utama penelitian ini dilakukan adalah untuk mengurangi ukuran sebuah *file* teks dengan memanfaatkan prinsip pengkodean Shannon-Fano. Metode penelitian yang digunakan meliputi langkah-langkah implementasi algoritma Shannon-Fano, yang melibatkan pembuatan tabel frekuensi kemunculan simbol dalam *file* teks dan pembangunan pohon Shannon-Fano. Teknik pemisahan kode diterapkan untuk membangun kode Shannon-Fano yang optimal berdasarkan frekuensi kemunculan simbol dalam *file* teks. Dalam metode penelitian ini juga dilakukan analisis kinerja algoritma Shannon-Fano berdasarkan faktor-faktor seperti kecepatan kompresi dan dekompresi, penggunaan memori, dan kompleksitas waktu. Data uji yang beragam telah digunakan untuk menguji kinerja algoritma, termasuk *file* teks dengan ukuran yang berbeda. Dari analisis kinerja yang dilakukan, didapatkan hasil yang menunjukkan bahwa algoritma Shannon-Fano memiliki kecepatan kompresi yang tinggi dan konsumsi memori yang efisien. Algoritma Shannon-Fano efektif dalam melakukan kompresi *file* teks dengan hasil yang signifikan. Ukuran *file* teks dapat dikurangi secara efisien tanpa mengorbankan keakuratan dekompresi. Hasil implementasi algoritma ini menunjukkan bahwa rata-rata ukuran *file* teks dapat berkurang sebesar 24.31%, dengan persentase compression ratio mencapai 75.68% dan ratio of compression sebesar 1.32. Hal ini menunjukkan bahwa algoritma Shannon-Fano dapat memberikan tingkat kompresi yang baik pada *file* teks.
Kata Kunci: *algoritma shannon-fano, compression, data processing, file teks*

1. Pendahuluan

Teknologi informasi dan komunikasi bergerak sangat cepat di dunia modern dalam berbagai bidang kehidupan. Sejumlah kemudahan diberikan, salah satunya adalah kemudahan data dari dokumen dapat ditulis dan disimpan secara digital (Prabiantissa et al., 2023.). Karena pesatnya perkembangan data dokumen yang semakin baru, Media penyimpanan dokumen digital tidak berbanding lurus dengan

fasilitas yang ada. Sementara itu, media penyimpanan yang tersedia masih sangat sedikit (Astawan, 2022).

Sebuah kajian ilmu komputer yang memiliki tujuan agar dapat mengurangi ukuran *file* sebelum disimpan atau dipindahkan ke perangkat penyimpanan (*storage device*) adalah kompresi data (Panjaitan et al., 2020). Kapasitas media penyimpanan seperti *compact disc* (CD) dan *harddisc* memiliki kapasitas terbatas. Media penyimpanan tidak dapat menyimpan data jika melebihi kapasitasnya dan jumlah data yang akan disimpan semakin bertambah. Akibatnya, kompresi data digunakan untuk mengatasi masalah ini (Zhang and Sun, 2021).

Kompresi data terdiri dari dua proses yaitu kompresi (*compression*) dan dekompresi (*decompression* atau pemulihan data ke keadaan semula). Jika *file* telah dikompresi, maka *file* tersebut harus bisa dibaca kembali setelah *file* tersebut di dekompresi. Gagasan di balik kompresi data adalah jika frekuensi karakter dalam kumpulan data diketahui, karakter tersebut dapat dikodekan untuk menghemat ruang penyimpanan data (Eko and Ningrum, 2019).

Algoritma yang umum digunakan adalah algoritma Shannon-Fano yang dikembangkan pada tahun 1948 oleh Claude Shannon dan Robert Fano. Algoritma ini membuat pohon kode biner alih-alih menggunakan kode dengan panjang tertentu (seperti kode ASCII). Memperoleh kode dengan angka atau bit yang lebih sedikit pada setiap karakter yang berisi data (Rada Kesuma, 2021). Shannon menjelaskan kode pendek ini, yang dikenal sebagai kode panjang variabel. Semakin sering kode muncul, semakin pendek kode tersebut, dan sebaliknya. Hal yang perlu diperhatikan dalam melakukan kompresi file adalah rasio kompresi (ukuran data setelah kompresi), optimalitas (apakah ukuran file sama dengan sebelum kompresi), dan kelengkapan (keutuhan data setelah kompresi) (Apriyanto, 2020).

Pada penelitian sebelumnya yang berjudul "Penerapan Metode *Shannon-Fano* Dalam Pengkompresian Data Teks" dan "Analisis Algoritma Shannon-Fano Dalam Kompresi Data Pengajuan Proposal Skripsi Mahasiswa STMIK CIC Cirebon", telah mampu memampatkan data untuk digunakan dalam untuk data *file* teks dengan menggunakan algoritma Shannon-Fano (Nas et al., 2019). Penelitian yang berjudul "Implementasi Algoritma Elias Gamma Kompresi Pada *File* Teks" telah menghasilkan sebuah aplikasi yang berhasil melakukan teknik kompresi dan dekompresi sebuah *file* teks (Nas et al., 2019).

Dari beberapa penelitian yang sudah dibahas sebelumnya, kompresi merupakan solusi yang tepat dalam mengatasi permasalahan tentang perkembangan era digitalisasi saat ini dimana banyaknya file yang disimpan dalam jumlah yang besar. Penelitian ini menggunakan algoritma Shannon-Fano yang memiliki performa yang baik dalam melakukan kompresi pada file teks.

2. Tinjauan Pustaka

2.1 Kompresi Data

Kompresi data merupakan sebuah proses mengubah masukan aliran data (*stream* sumber, atau juga dikenal sebagai data mentah asli) menjadi aliran data yang lebih kecil (bit *stream* hasil, atau juga dikenal sebagai *stream* yang sudah terkompresi) (Yuliastuti et al., 2023). Kompresi adalah sebuah ilmu komputer untuk memampatkan data sehingga hanya membutuhkan sedikit ruang untuk menyimpannya (Manalu, 2020). kompresi data bertujuan untuk menyampaikan data digital dengan sesedikit mungkin bit dalam bentuk teks, gambar, suara, dan media lainnya (Putro Utomo, 2020), namun akan tetap mempertahankan kebutuhan yang minimum untuk membentuk kembali data yang asli dengan sesedikit mungkin bit dalam bentuk teks, gambar, suara, dan media (Yanur Tanjung, 2021). Pada kompresi data terdapat dua teknik yang dapat diterapkan, yakni kompresi *lossy* dan *lossless* (Setiawan et al., 2023).

2.1.1 Kompresi *Lossy*

Adalah metode kompresi data yang menghasilkan *file* data terkompresi yang tidak dapat dipulihkan sebelum dikompresi sepenuhnya (Abdelwahab et al., 2021). Data yang didekodekan tidak dapat dikembalikan identik dengan aslinya saat data yang dikompresi didekodekan lagi. namun, ada bagian data yang hilang. Teknik ini biasa dipergunakan untuk sebuah data yang berupa grafik, *audio*, *video* dan yang lainnya.

2.1.2 Kompresi *Lossless*

Kompresi *Lossless* adalah kompresi yang membuat *file* data terkompresi yang dapat dikembalikan menjadi *file* data asli seperti sebelum dikompresi sepenuhnya tanpa perubahan apapun. Teknik ini biasanya digunakan untuk data berupa teks, zip, rar dan lain-lain.

2.2 Algoritma Shannon-Fano

Pada dasarnya algoritma Shannon-Fano mengubah setiap simbol data menjadi kode biner. Panjang kode kode biner ditentukan oleh probabilitas munculnya simbol. Algoritma Shannon-Fano mengkodekan setiap karakter dalam sekumpulan data masukan dengan sekumpulan beberapa bit. Karakter yang sering muncul dikodekan dengan rangkaian bit yang lebih pendek daripada karakter yang jarang muncul. Secara umum algoritma Shannon-Fano mencakup beberapa ketentuan yakni sebagai berikut:

1. Setiap karakter atau simbol yang beda mempunyai sebuah kode yang berbeda.
2. Karakter atau simbol dengan peluang kemunculan lebih tinggi mempunyai panjang kode yang lebih pendek begitupun simbol atau karakter dengan probabilitas yang lebih rendah memiliki bit kode lebih panjang.
3. Meski memiliki panjang bit berbeda dari kode karakter aslinya, kode yang dihasilkan dapat dikodekan secara unik.

Berikut ini merupakan langkah-langkah kompresi dengan menggunakan algoritma Shannon-Fano, antara lain:

1. Buatlah pohon dengan membaca semua karakter dalam teks dan menghitung probabilitas kemunculan setiap karakter.
2. Karakter-karakter ini diurutkan dari karakter yang peluang kemunculannya paling tinggi hingga karakter yang peluang kemunculannya paling rendah.
3. Pisahkan kedua rangkaian karakter ini menjadi dua himpunan bagian dengan jumlah probabilitas yang sama atau hampir sama.
4. Semua simbol pada himpunan bagian pertama diberi kode 0, dan simbol pada himpunan bagian lainnya diberi kode 1.
5. Jika hanya ada dua jenis simbol dalam suatu subset, kodenya dibedakan dengan memberikan bit berbeda pada setiap simbol.
6. Ulangi langkah 3, 4, dan 5 secara rekursif untuk setiap subset hingga tidak ada lagi subset.

Berikut ini merupakan langkah-langkah algoritma untuk melakukan dekompresi pada serangkaian kode-kode tersebut, antara lain:

1. Baca bit pertama dari rangkaian kode yang dihasilkan.
2. Jika bit ada di pohon, bit tersebut dikonversi ke karakter yang sesuai dengan bit tersebut.
3. Jika bit tersebut tidak ada di pohon, gabungkan dengan bit berikutnya dalam rantai kode dan periksa dengan tabel hasil yang dikodekan.
4. Lakukan langkah 3 hingga Anda menemukan string bit yang cocok dengan pohon, ubah string bit menjadi karakter yang sesuai.
5. Baca bit berikutnya dan ulangi langkah 2, 3, dan 4 hingga urutan kode selesai.

2.3 Rasio Kompresi

Rasio kompresi adalah tingkat pengurangan data sebagai hasil dari proses kompresi. Rasio merupakan pembanding antara panjang *string* data terkompresi dan panjang *string* data asli, Berikut ini adalah beberapa parameter umum yang digunakan untuk menggambarkan seberapa baik metode kompresi bekerja (Apriyanto, 2020):

1. *Compression Ratio (CR)*

   Compression Ratio (CR) adalah persentase besar data yang telah dikompresi yang didapat dari hasil perbandingan antara ukuran data setelah dikompresi dengan ukuran data sebelum dikompresi. Secara matematis dapat ditulis seperti pada persamaan 2.1:

$$CR = \frac{Ukuran\ data\ setelah\ dikompresi}{Ukuran\ data\ sebelum\ Dikompresi} \times 100\% \qquad (1)$$

 ISSN: 2579-566X (Online)
ISSN: 2477-5274 (Print)

2. *Ratio Of Compression (RC)*

   Ratio of Compression (RC perbandingan antara ukuran data sebelum dikompresi dengan ukuran data setelah dikompresi. Secara matematis dapat ditulis seperti pada persamaan 2.2:

$$RC = \frac{Ukuran\ data\ sebelum\ dikompresi}{Ukuran\ data\ sesudah\ Dikompresi} \quad (2)$$

3. *Space Savings (SS)*

   Space Savings (SS) adalah menghitung persentasi penyusutan dari data sebelum dilakukan kompresi. Secara matematis dapat ditulis seperti pada persamaan 2.3:

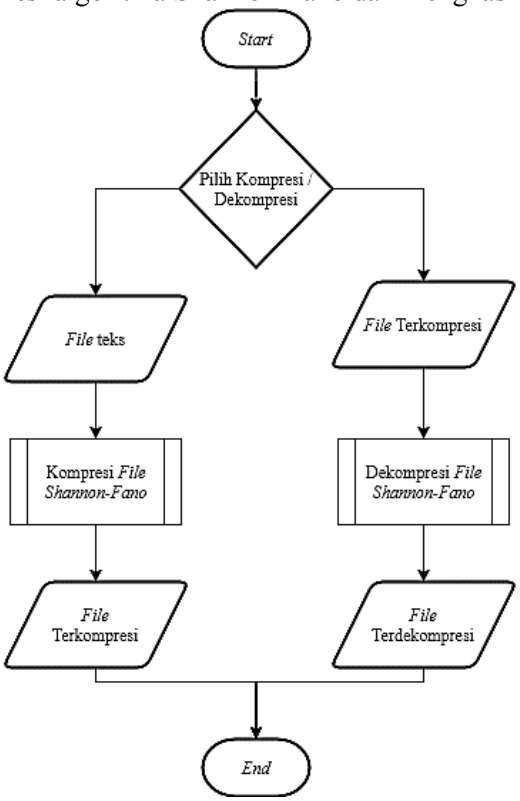
$$SS = 1 - CR \quad (3)$$

4. Waktu Kompresi

   Jumlah waktu yang dibutuhkan sistem untuk memasukkan *file* teks yang perlu dikompresi sebelum proses kompresi selesai dikenal sebagai waktu kompresi. Teknik kompresi yang digunakan semakin efisien maka semakin cepat sistem melakukan kompresi.

3. Metode Penelitian

Dalam dilakukannya penelitian tentang kompresi data, Penulis membuat gambaran umum tentang sistem yang akan dibuat. Pada gambar 1 menerangkan bahwa sistem yang akan dibuat oleh penulis yaitu tentang kompresi data dengan jenis *lossless*. dimana kompresi yang membuat *file* data terkompresi yang dapat dikembalikan menjadi *file* data asli seperti sebelum dikompresi sepenuhnya tanpa perubahan apapun. Pada menu kompresi hal pertama yang dilakukan yaitu menginputkan *file* teks lalu kompresi diproses dengan algoritma Shannon-Fano dan menghasilkan *file* terkompresi. Selanjutnya pada menu dekompresi, *file* terkompresi tersebut dapat dikembalikan seperti aslinya dengan menggunakan proses dekompresi algoritma Shannon-Fano dan menghasilkan *file* terdekompresi.



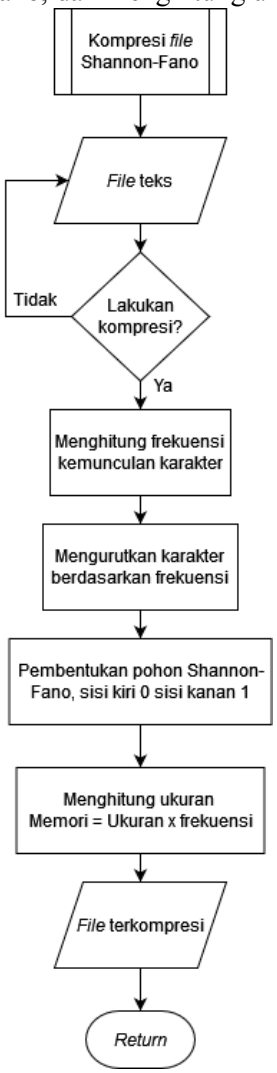
Gambar 1. Flowchart Sistem Kompresi dan Dekompresi

Sistem yang akan dibuat oleh penulis yaitu tentang kompresi data dengan jenis *lossless.* dimana kompresi yang membuat *file* data terkompresi yang dapat dikembalikan menjadi *file* data asli seperti sebelum dikompresi sepenuhnya tanpa perubahan apapun. Pada menu kompresi hal pertama yang dilakukan yaitu menginputkan *file* teks lalu kompresi diproses dengan algoritma Shannon-Fano dan

menghasilkan *file* terkompresi. Selanjutnya pada menu dekompresi, *file* terkompresi tersebut dapat dikembalikan seperti aslinya dengan menggunakan proses dekompresi algoritma Shannon-Fano dan menghasilkan *file* terdekompresi.

3.1 Kompresi Menggunakan Algoritma Shannon-Fano

Pada proses kompresi file dengan menggunakan algoritma Shannon-fano, ada beberapa proses yang dilakukan yaitu menghitung frekuensi kemunculan karakter, mengurutkan karakter berdasarkan frekuensi, membentuk pohon Shannon fano, dan menghitung ukuran memori.

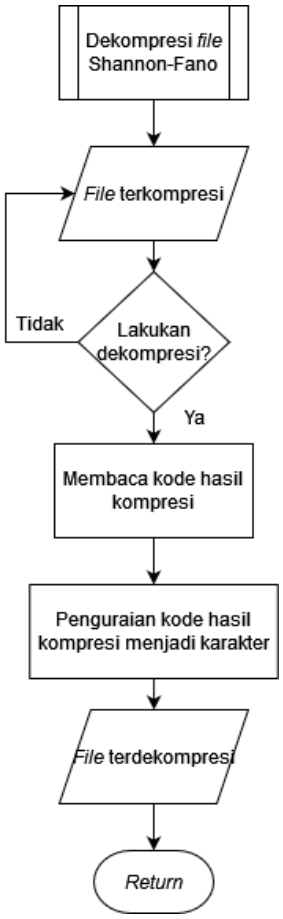


Gambar 1. Alur Kompresi Algoritma Shannon-Fano

3.2 Dekompresi menggunakan Algoritma Shannon-Fano

Dalam dekompresi hal pertama yang dilakukan yaitu menginputkan sebuah *file* hasil dekompresi ke program dekompresi. Jika sudah proses yang akan berjalan sesuai langkah-langkah dari algoritma Shannon-Fano yaitu yang pertama membaca bit kode hasil kompresi lalu menguraikannya menjadi karakter. Setelah itu akan didapatkan sebuah *file* hasil dekompresi.

 ISSN: 2579-566X (Online)
ISSN: 2477-5274 (Print)



Gambar 3. Alur Dekompresi Algoritma Shannon-Fano

4. Hasil dan Pembahasan

4.1 Pengujian Kompresi

Penulis menggunakan 4 parameter untuk menyatakan kinerja metode kompresi (*Compression Ratio, Ratio Of Compression, Space Savings*, Waktu Kompresi). Dengan menggunakan parameter tersebut nantinya pengujian kompresi menghasilkan sebuah tabel 1.

Tabel 1. Pengujian Ukuran File .txt Hasil Kompresi

No	*Nama dan Jenis File*	*Ukuran File Sebelum Kompresi (KB)*	*Ukuran File Sesudah Kompresi (KB)*	*Waktu Kompresi*
1	Data_Uji_01.txt	100	79	0.19
2	Data_Uji_02.txt	300	236	0.41
3	Data_Uji_03.txt	500	400	0.63
4	Data_Uji_04.txt	800	611	0.75
5	Data_Uji_05.txt	1000	731	1.08

Dengan informasi dalam tabel 1, kita dapat melihat seberapa efektif algoritma kompresi yang digunakan dalam mengurangi ukuran *file*. Semakin besar perbedaan antara ukuran sebelum dan setelah kompresi, semakin efektif algoritma kompresi tersebut. Selain itu, kita juga dapat melihat waktu yang dibutuhkan untuk proses kompresi, yang dapat memberikan gambaran tentang kecepatan algoritma kompresi yang digunakan.

Tabel 2. Pengujian Frekuensi, Rasio, Space Savings file.txt

No	*Nama dan Jenis File*	*Frekuensi Kemunculan*	*Ratio of Compression*	*Compression Ratio (%)*	*Space Savings (%)*
1	Data_Uji_01.txt	101167	1.27	79.0	21.0
2	Data_Uji_02.txt	302002	1.27	78.6	21.4
3	Data_Uji_03.txt	504874	1.25	79.96	20.04
4	Data_Uji_04.txt	804160	1.31	76.35	23.65
5	Data_Uji_05.txt	1011237	1.37	73.0	27.0

Dengan informasi dalam tabel 2, kita dapat melihat seberapa efektif algoritma kompresi dalam mengurangi ukuran *file* dengan memperhatikan frekuensi kemunculan karakter atau elemen tertentu. Selain itu, kita juga dapat melihat persentase kompresi dan penghematan ruang yang dicapai oleh algoritma kompresi tersebut.

4.2 Pengujian Dekompresi

Pengujian dekompresi dilakukan penulis untuk mengetahui ukuran *file* sebelum dan sesudah didekompresi ada menguji isi *file* setelah dekompresi seperti pada tabel 3.

Tabel 3. Pengujian File .txt Hasil Dekompresi

No	*Nama dan Jenis File*	*Ukuran File Asli (KB)*	*Ukuran File Sesudah Kompresi (KB)*	*Isi File Setelah Kompresi*
1	Decompressed_01.txt	100	79	100
2	Decompressed_02.txt	300	236	300
3	Decompressed_03.txt	500	400	500
4	Decompressed_04.txt	800	611	800
5	Decompressed_05.txt	1000	731	1000

Tabel 3 memberikan gambaran tentang bagaimana proses dekompresi telah mengembalikan *file* ke bentuk aslinya dengan ukuran yang sama atau mirip dengan ukuran *file* asli. Ini menunjukkan bahwa proses dekompresi telah berhasil mengembalikan *file* ke keadaan semula tanpa kehilangan informasi yang signifikan.

5. Kesimpulan

Algoritma Shannon-Fano dapat digunakan untuk melakukan kompresi data dengan efektif jika *file* yang dikompresi berjenis .txt. Rata-rata yang dihasilkan dari *compression ratio* sebesar 75,68% dan *ratio of compression* sebesar 1,32. Tingkat kompresi yang dicapai oleh algoritma Shannon-Fano dapat bervariasi tergantung pada panjang teks yang dikompresi. Rata-rata yang dihasilkan dari space savings atau penyusutan data sebesar 24.31%. Waktu yang dibutuhkan oleh algoritma Shannon-Fano untuk melakukan kompresi dapat bervariasi tergantung pada ukuran dan kompleksitas data teks yang dikompresi. Pada pengujian yang telah dilakukan maka dihasilkan waktu tercepat 0.19 detik dan yang terlama 83.19 detik.

Referensi

Abdelwahab, O.F., Hussein, A.I., Hamed, H.F.A., Kelash, H.M., Khalaf, A.A.M., 2021. Efficient Combination of RSA Cryptography, Lossy, and Lossless Compression Steganography Techniques to Hide Data, in: Procedia Computer Science. Elsevier B.V., pp. 5–12.

Chairun Nas, Wanda Ilham, & Ilwan Syafrinal. (2019). *Analisis Algoritma* Shannon-Fano *Dalam Kompresi Data Pengajuan Proposal Skripsi Mahasiswa STMIK CIC Cirebon. Vol. 8, No. 2*, 91–100.

Citra Nurina Prabiantissa, Danang Haryo Sulaksono, Gusti Eka Yuliastuti, & Adjie Prasetyo Nugroho. (2023). *Implementasi Algoritma Kompresi Lempel-Ziv-Welch pada Data Citra*. 321–325. https://doi.org/10.31284/p.snestik.2023.4344

Febika Rada Kesuma. (2021). *Implementasi Kombinasi Algoritma Elias Omega Codes Dan Golomb Rice Dengan Teknik High Compress Pada File Teks*. *Vol 8 No 2*, 76–81.

Gyanti Eko Retno Ningrum. (2019). *Penerapan Metode Shanno Fano Dalam Pengkompresian Data Teks*. *Vol. 6, No. 2*, 229–234.

I Kadek Todi Astawan. (2022). *Implementasi E-Filling System sebagai Pusat Penyimpanan Data Perusahaan Medi Groups berbasis Cloud dengan menggunakan Aplikasi Google One*. *Vol. 1, No. 2*, 1–11.

Muhammad Apriyanto & Hutrianto. (2021). *Analisa Penerapan Algortima Goldbach Codes Dan Metode* Shannon-Fano *Pada Kompresi File Teks*. *Vol. 2, No 5*, 207–218.

Rizki Yanur Tanjung & Mesran. (2021). *Perancangan Aplikasi Kompresi File Dokumen Menggunakan Algoritma Adiitive Code. Vol. 8 No. 4,* 108−113.

Rosalina Manalu. (2020). *Implementasi Algoritma Massey-Omura dan Algoritma Elias Delcta Code Pada Pengamanan dan Kompresi File Dokumen*. *Vol. 1 No. 3*, 239–246. https://doi.org/DOI 10.30865/json.v1i3.2152

Setiawan, M. C., Yuliastuti, G. E., Rachman, A., Adhi, I. T., & Surabaya, T. (2023). Implementasi Metode Lempel-Ziv-Welch pada Kompresi File Teks. *Seminar Nasional Sains Dan Teknologi Terapan XI 2023 Institut Teknologi Adhi Tama Surabaya*.

Sinta M. Panjaitan, Surya Darma Nasution, & Bister Purba. (2020). *Penerapan Algoritma Gopala-Hemachandra Code2 (GH-2(n)) Pada Kompresi File Audio*. *Vol 4, No 1*, 170–177.

Soumi Rohmah Saragih & Dito Putro Utomo. (2020). *Penarapan Algoritma Prefix Code Dalam Kompresi Data Teks*. *Vol 4, No 1*, 249–252.

Yuliastuti, G. E., Sulaksono, D. H., Prabiantissa, C. N., & Febrianto, A. (2023). Implementasi Algoritma Levenshtein untuk Kompresi File Audio. *Seminar Nasional Teknik Elektro, Sistem Informasi, Dan Teknik Informatika (SNESTIK)*, 314–320. https://ejurnal.itats.ac.id/snestik https://snestik.itats.ac.id

Zhang, J., Sun, D., 2021. Improvement of data compression technology for power dispatching based on run length encoding, in: Procedia Computer Science. Elsevier B.V., pp. 526–532.