Management of Information System Journal
Vol 4, No 2, Maret 2026, Hal. 87 - 95
ISSN: 2964-9455 (media online)
DOI: 10.47065/mis.v2i3.2568
https://journal.fkpt.org/index.php/MIS

Analisa Perbandingan Algoritma Shannon Fano Dan Algoritma Stout Code Pada Kompresi File Teks

Dito Putro Utomo[1,*], Abdul Karim[2], Muhammad Syahrizal[3]

[1]Jurusan Teknik Komputer dan Informatika, Program Studi Teknologi Rekayasa Multimedia Grafis, Politeknik Negeri Medan, Medan, Indonesia
[3]Fakultas Sains dan Teknologi, Program Studi Teknologi Informasi, Universitas Labuhanbatu, Rantauparapat, Indonesia
[2]Program Studi Teknologi Rekayasa Komputer Grafis, Politeknik Cendana, Medan, Indonesia
Email: [1,*]ditoputroutomo@polmed.ac.id, , [2]Abdulkarim@ulb.ac.id, [3]m.syahrizal@politeknikcendana.ac.id
Email Penulis Korespondensi: ditoputroutomo@polmed.ac.id

Abstrak−Perkembangan teknologi yang pesat saat ini banyak menarik perhatian masyarakat luas, perkembangan komputer yang dinamis diiringi dengan mendapatkan informasi yang sangat cepat. Kompresi data adalah teknik untuk mengurangi jumlah data dari data aslinya. Kompresi data biasanya diterapkan pada mesin komputer. Hal ini terjadi karena setiap simbol yang ditampilkan pada komputer memiliki nilai bit yang berbeda. Ukuran file teks yang yang besar akan menjadi masalah pada ruang penyimpanan. Karena kebutuhan akan file teks sangat penting, kita cenderung mengumpulkan data berupa file teks dan seringkali tanpa disadari kita sering menyimpan dalam ukuran besar, hal ini menyebabkan kebutuhan media penyimpan tidak sedikit. Untuk mengatasi masalah tersebut digunakan file teks yang memiliki ukuran lebih besar dengan melakukan kompresi file teks. Data yang berukuran besar akan dikompresi menjadi ukuran yang kecil dan akan mengurangi penyimpanan. Setelah menerapkan perbandingan Algoritma Shannon Fano dan algoritma Stout Code mengkompresi file teks telah membuktikan bahwa file teks tersebut telah berhasil dikompresi. Setelah melakukan proses kompresi file teks penulis dapat disimpulkan bahwa Algoritma Shannon Fano lebih baik dalam melakukan proses kompresi.

Kata Kunci: Kompresi; Algoritma Shannon Fano; Algoritma Stout Code; File Teks

Abstract−The rapid development of technology today attracts a lot of attention from the wider community. The dynamic development of computers is accompanied by the ability to get information very quickly. Data compression is a technique to reduce the amount of data in the original data. Data compression is usually applied to computer machines. This happens because each symbol displayed on the computer has a different bit value. The large size of text files will be a problem for storage space. Because the need for text files is very important, we tend to collect data in the form of text files, and often without realizing it, we store it in large sizes. This causes the need for storage media to be large. To overcome this problem, text files that have a larger size are used by compressing text files. Large data will be compressed into a small size, which will reduce storage. After applying the comparison of the Shannon Fano Algorithm and the Stout Code algorithm, compressing the text file has proven that the text file has been successfully compressed. After performing the text file compression process, the author can conclude that the Shannon Fano algorithm is better at performing the compression process.

Keywords: Compression; Shannon Fano Algorithm; Stout Code Algorithm; Text Files

1. PENDAHULUAN

Dengan seiring berkembangnya kemajuan teknologi yang begitu cepat hingga saat ini sangat berguna dalam mengetahui informasi secara global yang dapat meliputi dari berbagai teks, gambar, video, dan audio. File teks adalah yang berisi data dalam benuk teks. Data yang diperoleh berasal dari dokumen pengolah kata di mana banyak di simpan manusia di komputer mereka masing-masing, dan ditambah lagi dengan perusahaan yang mempunyai banyak dokumen dan pastinya perusahaan akan membutuhkan kapasitas penyimpanan yang cukup besar juga[1]. Perkembangan teknologi yang semakin maju dan penambahan penguna komputer yang semakin banyak menyebabkan menumpuknya data serta perpindahan data dari satu perangkat ke perangkat lain. Data-data tersebut umumnya dikompresikan terlebih dahulu, agar proses penyimpanan atau pertukaran data tidak memakan waktu terlalu lama[2].

Kompresi merupakan suatu proses yang dapat mengubah sebuah aliran data sehingga ukurannya menjadi lebih kecil dari ukuran sebelumnya. Proses ini digunakan untuk cabang dibidang teknik. Proses kompresi dilakukan untuk berbagai tujuan, untuk menghasilkan data menjadi lebih kecil dalam proses pengiriman dan perpindahan data[3], [4], [5]. Kompresi data adalah sebuah cara untuk memperkecil data sehingga memerlukan ruangan penyimpanan lebih kecil dan lebih efesien dalam menyimpannya atau mempersingkat waktu pertukaran data. Ada dua teknik yang digunakan dalam kompresi data yaitu, kompresi tanpa kehilangan data (lossless compression) dan kompresi data berkehilangan (lossy compression)[6], [7], [8].

Dalam keterbatasan file teks yang memiliki ukuran data yang besar dan akan menjadi masalah pada ruang penyimpanan. Karena kebutuhan akan file teks yang sangat penting, maka manusia cenderung mengumpulkan data berupa file teks dan seringkali tanpa disadari manusia menyimpan dan mengirimkannya dalam ukuran yang besar dan jelas, hal ini menyebabkan kebutuhan akan media penyimpanan yang tidak sedikit[9]. Apalagi saat ini manusia lebih sering melakukan transfer file teks baik secara online maupun offline yang tentunya akan membutuhkan waktu pengiriman yang cepat dan juga ruang penyimpanan yang memadai. Adapun solusi untuk mengatasi masalah tersebut adalah dengan melakukan kompresi. Hal ini bertujuan agar ukuran file teks menjadi jauh lebih kecil sehingga proses pengiriman file teks lebih cepat serta dapat menghemat ruang penyimpanan[10].

Management of Information System Journal
Vol 4, No 2, Maret 2026, Hal. 87 - 95
ISSN: 2964-9455 (media online)
DOI: 10.47065/mis.v2i3.2568
https://journal.fkpt.org/index.php/MIS

Berbagai algoritma kompresi telah dikembangkan untuk mengatasi permasalahan tersebut, di antaranya algoritma Shannon-Fano yang merupakan metode klasik berbasis probabilitas kemunculan simbol, serta algoritma Stout Code yang menawarkan pendekatan alternatif dalam pembentukan kode[11], [12]. Namun demikian, sebagian besar penelitian yang ada lebih berfokus pada perbandingan algoritma Shannon-Fano dengan algoritma yang lebih populer seperti Huffman, sehingga kajian yang secara khusus membandingkan Shannon-Fano dengan Stout Code masih sangat terbatas. Selain itu, belum banyak penelitian yang menganalisis bagaimana kedua algoritma tersebut bekerja pada berbagai karakteristik file teks, terutama yang memiliki distribusi frekuensi karakter yang berbeda-beda. Kondisi ini menimbulkan permasalahan terkait kurangnya informasi empiris mengenai keunggulan dan kelemahan masing-masing algoritma dalam konteks kompresi teks, baik dari segi rasio kompresi, efisiensi waktu, maupun kompleksitas proses[13], [14], [15], [16].
Urgensi penelitian ini terletak pada pentingnya menyediakan dasar analisis yang lebih komprehensif dalam menentukan algoritma kompresi yang tepat sesuai dengan kebutuhan dan karakteristik data. Di tengah meningkatnya volume data teks seperti dokumen digital, log sistem, dan konten berbasis teks lainnya, pemilihan algoritma kompresi yang efisien menjadi sangat krusial untuk mengoptimalkan penggunaan ruang penyimpanan dan mempercepat proses transmisi data. Selain itu, penelitian ini juga memiliki nilai ilmiah dalam memperkaya literatur mengenai algoritma kompresi yang kurang populer seperti Stout Code, serta mengevaluasi kembali relevansi algoritma klasik seperti Shannon-Fano dalam konteks teknologi saat ini. Dengan demikian, hasil penelitian diharapkan dapat memberikan kontribusi baik secara teoritis maupun praktis dalam pengembangan dan pemilihan metode kompresi data teks yang lebih optimal.
Ada beberapa algoritma dalam melakukan kompresi pada file teks, namun pada penelitian ini menggunakan algoritma Shannon Fano dan algoritma Stout Code. Alasan menggunakan algoritma tersebut karena algoritma Shannon Fano dan Stout Code merupakan jenis kompresi Lossless Compression, dimana data hasil kompresi bisa dikembalikan pada data awal sebelum dilakukannya proses kompresi.
Beberapa penelitian seperti yang dilakukan oleh Eko Priyono dan Hindayati Mustafidah pada tahun 2024 dengan judul penelitian Text Compression Using the Shannon-Fano, Huffman, and Half-Byte Algorithms dimana hasil yang didapatkan bahwa algoritma Shannon-Fano memiliki compression ratio sebesar 40.36%, lebih rendah dibandingkan Huffman yang mencapai 46.05%, namun jauh lebih baik dibandingkan Half-Byte. Kesimpulan dari penelitian ini adalah bahwa Shannon-Fano cukup efektif untuk kompresi teks, tetapi tidak optimal karena tidak selalu menghasilkan kode dengan panjang minimum[17].
Penelitian lainnya yang dilakukan oleh Mohammed Abdullah Al-Habib dan Ahmed Hassan Al-Saadi pada tahun 2021 dengan judul penelitian A Comparative Study of Data Compression Algorithms for Text Data dimana hasil dari penelitian Shannon-Fano dibandingkan dengan beberapa algoritma lain seperti Huffman dan Run-Length Encoding. Hasilnya menunjukkan bahwa Shannon-Fano memiliki performa kompresi yang cukup stabil, tetapi kalah efisien dibandingkan Huffman dalam hal rasio kompresi dan ukuran output akhir[18].
Selain itu, penelitian juga dilakukan oleh Anjali Sharma dan Rajesh Kumar Singh pada tahun 2022 dengan judul penelitian Performance Analysis of Lossless Text Compression Techniques Using Shannon-Fano and Huffman Coding dimana hasil yang didapatkan bahwa Shannon-Fano memiliki struktur algoritma yang lebih sederhana, tetapi dari sisi efisiensi kompresi dan panjang kode, Huffman tetap lebih unggul. Shannon-Fano cenderung menghasilkan ukuran file yang sedikit lebih besar karena pembagian probabilitas yang tidak selalu optimal[19].
Dan penelitian yang dilakukan oleh Sandeep Kumar Verma dan Neha Gupta pada tahun 2023 dengan judul penelitian An Efficient Lossless Data Compression Technique Using Variable Length Coding dimana dari proses penelitian didapatkan hasil pengembangan teknik kompresi berbasis variable-length coding yang memiliki prinsip serupa dengan Shannon-Fano dan Stout Code. Hasil penelitian menunjukkan bahwa pendekatan variable-length modern mampu meningkatkan compression ratio dan efisiensi waktu dibandingkan metode klasik, sehingga membuka peluang pengembangan lebih lanjut pada algoritma seperti Stout Code[20].
Perbedaan utama penelitian ini dibandingkan dengan penelitian terdahulu terletak pada objek perbandingan yang digunakan, yaitu algoritma Shannon-Fano dan Stout Code, di mana penelitian sebelumnya umumnya hanya membandingkan Shannon-Fano dengan algoritma yang lebih umum seperti Huffman. Selain itu, penelitian ini secara khusus memfokuskan pada penggunaan Stout Code yang masih jarang diteliti dalam kompresi file teks, sehingga memberikan kontribusi baru dalam pengembangan algoritma kompresi data. Dari sisi metodologi, penelitian ini juga menggunakan parameter evaluasi yang lebih komprehensif, tidak hanya terbatas pada compression ratio, tetapi juga mencakup waktu proses dan efisiensi algoritma. Dengan demikian, penelitian ini diharapkan mampu memberikan analisis yang lebih mendalam serta memperkaya kajian ilmiah terkait perbandingan algoritma kompresi lossless, khususnya antara metode klasik dan metode yang masih belum banyak dieksplorasi.

2. METODOLOGI PENELITIAN

2.1 Kompresi

Kompresi adalah mengubah data yang berupa kumpulan karakter menjadi bentuk kode dengan tujuan untuk menghemat kebutuhan tempat ruang penyimpanan dan waktu transmisi data. Saat memilih algoritma yang akan digunakan untuk kompresi data file teks, beberapa faktor dipertimbangkan, termasuk sumber daya yang diperlukan (memori, kecepatan PC), kecepatan kompresi, ukuran hasil kompresi, besarnya redundansi, dan komplesitas

Management of Information System Journal
Vol 4, No 2, Maret 2026, Hal. 87 - 95
ISSN: 2964-9455 (media online)
DOI: 10.47065/mis.v2i3.2568
https://journal.fkpt.org/index.php/MIS

algoritma[21], [22]. Jika semakin besar ukuran data yang disimpan maka membutuhkan media penyimpanan yang besar. Oleh karena itu, kemudian muncul metode-metode yang bertujuan untuk mengkompresi data agar dapat menghemat tempat penyimpanan data. Salah satunya yaitu kompresi data teks agar bisa diperkecil ukurannya[23], [24].

2.2 File Text

File teks adalah file yang berisi data dalam bentuk teks. Data yang diperoleh berasal dari dokumen pengolah kata, bilangan angka yang digunakan untuk melakukan perhitungan, serta menginput nama dan alamat dalam database merupakan contoh input data teks yang terdiri dari karakter, angka dan huruf[25], [26]. Fungsi file teks yaitu untuk menyimpan data atau informasi yang dapat digunakan dan dikelola oleh user, jika semakin banyak file teks yang akan disimpan, maka semakin banyak pula ruang yang akan digunakan. Maka daripada itu diperlukan suatu metode untuk mengecilkan ukuran file teks supaya lebih sedikit menggunakan ruang penyimpanan yang biasa disebut dengan kompresi.

2.3 Algoritma Shannon-Fano

Algoritma Shannon-Fano coding ditemukan oleh Claude Shannon (bapak teori informasi) dan Robert Fano pada tahun 1949. Pada saat itu metode ini merupakan metode yang paling baik tetapi hampir tidak pernah digunakan dan dikembangkan lagi setelah kemunculan algoritma Huffman. Pada dasarnya metode ini menggantikan setiap simbol dengan alternative kode biner yang panjangnya ditentukan berdasarkan probabilitas dari simbol tersebut. Adapun langkah-langkah kompresi pada Algoritma Shannon-Fano adalah sebagai berikut[27], [28]:
Proses pengkodean dimulai dengan menyusun daftar peluang atau frekuensi kemunculan setiap karakter dari pesan yang akan dikodekan, kemudian daftar tersebut diurutkan secara menurun berdasarkan frekuensi kemunculannya, mulai dari yang paling sering hingga yang paling jarang muncul. Selanjutnya, daftar ini dibagi menjadi dua bagian dengan tujuan agar total frekuensi pada masing-masing bagian seimbang atau sedekat mungkin, di mana bagian pertama disebut bagian atas dan bagian kedua disebut bagian bawah. Setelah pembagian dilakukan, setiap karakter dalam bagian atas diberikan awalan kode tertentu (misalnya 0), sedangkan karakter pada bagian bawah diberikan awalan kode lain (misalnya 1). Proses ini kemudian dilanjutkan secara rekursif pada masing-masing bagian dengan cara membagi kembali setiap kelompok menjadi subkelompok yang lebih kecil berdasarkan keseimbangan frekuensi, sambil menambahkan bit kode pada setiap tahap, hingga akhirnya setiap karakter memperoleh representasi kode biner yang unik sesuai dengan struktur pembagian yang terbentuk.

2.4 Algoritma Stout Code

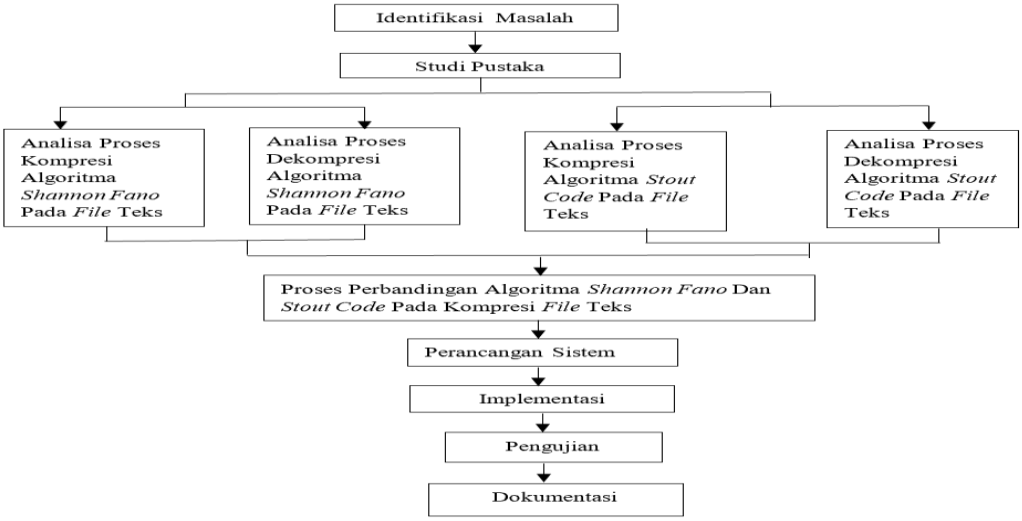
Algoritma Stout code adalah algoritma kompresi yang mengkompresi data ke dalam bentuk data lain dengan menggunakan bit yang lebih sedikit dan bit karakter yang lebih banyak. Penggunaan algoritma Stout Code pada aplikasi kompresi file teks adalah untuk mengubah ukuran data asli menjadi ukuran data yang lebih kecil. Prosedur kompresi data teks diawali dengan pemilihan data teks yang akan dikompres, data tersebut dikompres menggunakan algoritma Stout Code sehingga menghasilkan data terkompresi dengan ukuran yang lebih kecil. Pada proses dekompresi, keluaran berupa data terkompresi dan data tersebut didekompresi menggunakan algoritma Stout Code untuk menghasilkan file teks dengan ukuran yang sebelumnya dikompresi atau diubah menjadi data awal. Langkah-langkah kompresi algoritma Stout Code adalah sebagai berikut[29], [30]
Proses kompresi data dimulai dengan menyiapkan file teks yang akan dikompresi, kemudian file tersebut dibuka menggunakan software HxD untuk memperoleh representasi nilai dalam bentuk hexadecimal. Dari hasil tersebut diambil sebanyak 20 nilai hexadecimal sebagai sampel, lalu disusun dalam bentuk tabel untuk memudahkan analisis. Nilai-nilai hexadecimal tersebut kemudian diurutkan berdasarkan frekuensi kemunculannya, di mana nilai dengan frekuensi tertinggi ditempatkan pada urutan teratas. Setelah itu dibuat tabel nilai hexadecimal sebelum dikompresi untuk mengetahui total jumlah bit awal sebagai acuan perbandingan. Tahap berikutnya adalah melakukan kompresi menggunakan algoritma Stout Code dengan cara menggantikan representasi biner setiap nilai menjadi kode S2(n). Hasil kompresi ini kemudian disusun dalam tabel nilai hexadecimal yang telah dikodekan untuk melihat perubahan jumlah bit setelah proses kompresi. Selanjutnya, string bit hasil kompresi disusun kembali sesuai urutan posisi nilai hexadecimal awal, lalu dibuat tabel string bit berdasarkan codeword S2(n) yang telah terbentuk. Setelah itu dilakukan penambahan padding dan flagging untuk menyesuaikan panjang bit, kemudian seluruh string bit dibagi menjadi kelompok per 8 bit. Hasil pengelompokan ini ditampilkan dalam bentuk tabel, lalu setiap kelompok biner dikonversi kembali menjadi nilai hexadecimal untuk mengetahui karakter yang dihasilkan berdasarkan kode ASCII. Tahap akhir adalah menyusun tabel hasil karakter terkompresi sebagai representasi akhir dari proses kompresi data yang telah dilakukan.

2.5 Tahapan Penelitian

Kerangka kerja ini merupakan prosedur yang akan diambil untuk memecahkan masalah yang terdiri dari beberapa tahapan yang saling terkait dan akan dibahas. Kerangka kerja ini diperlukan untuk memudahkan dalam suatu penelitan. Berikut erangka penelitian, maka pembahasan masing-masing tahapan penelitian dapat diuraikan sebagai berikut:
Penelitian ini diawali dengan tahap identifikasi masalah untuk menentukan algoritma kompresi yang paling efektif antara Shannon Fano dan Stout Code dalam memperkecil ukuran file teks. Selanjutnya dilakukan studi pustaka dengan mengkaji berbagai sumber seperti buku, jurnal, dan artikel ilmiah guna memperoleh landasan teori yang relevan. Tahap berikutnya adalah analisis proses kompresi pada file teks menggunakan kedua algoritma tersebut untuk memahami mekanisme dan

Management of Information System Journal
Vol 4, No 2, Maret 2026, Hal. 87 - 95
ISSN: 2964-9455 (media online)
DOI: 10.47065/mis.v2i3.2568
https://journal.fkpt.org/index.php/MIS

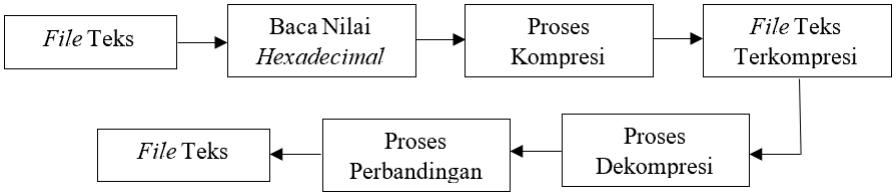
efisiensi masing-masing metode dalam mereduksi ukuran data. Setelah itu dilakukan analisis proses dekompresi guna memastikan bahwa data yang telah dikompresi dapat dikembalikan ke bentuk semula tanpa kehilangan informasi. Proses perbandingan kemudian dilakukan untuk mengevaluasi kinerja algoritma Shannon Fano dan Stout Code berdasarkan hasil kompresi yang diperoleh. Tahap selanjutnya adalah perancangan sistem yang bertujuan untuk mengimplementasikan hasil analisis dalam bentuk aplikasi perbandingan kompresi file teks menggunakan bahasa pemrograman Microsoft Visual Basic .NET 2008. Setelah perancangan, dilakukan tahap implementasi untuk membangun perangkat lunak sesuai dengan desain yang telah dibuat, kemudian dilanjutkan dengan tahap pengujian guna memastikan sistem berjalan dengan baik dan menghasilkan output yang sesuai, khususnya dalam membandingkan kinerja kedua algoritma pada file teks berformat docx. Tahap akhir adalah dokumentasi, yaitu penyusunan laporan penelitian yang menjelaskan seluruh proses dan hasil yang diperoleh, sehingga dapat menjadi referensi bagi pengembangan penelitian atau aplikasi selanjutnya.
Adapun kerangka kerja penelitian yang telah diuraikan di atas dapat digambarkan pada gambar 3.1 sebagai berikut:



Gambar 1. Kerangka Kerja Penelitian

3. HASIL DAN PEMBAHASAN

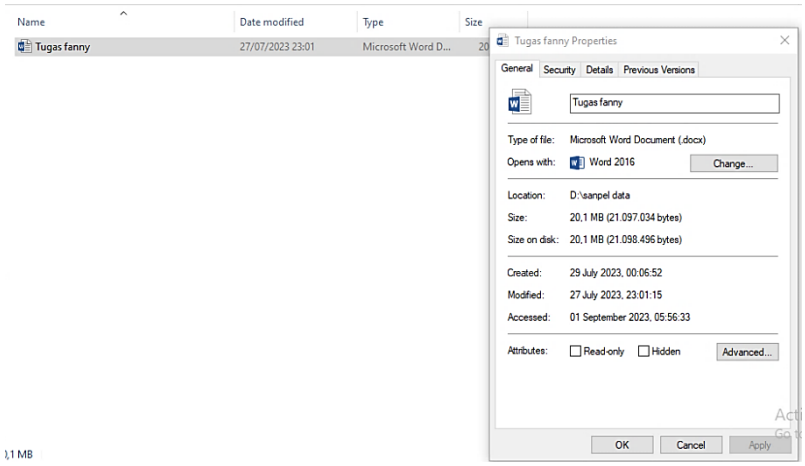
Analisa permasalahan yang dibahas dalam penelitian ini adalah bagaimana kinerja algoritma Shannon Fano dan algoritma Stout Code dalam melakukan kompresi file teks dan perbandingan kinerja kedua algoritma tersebut berdasarkan parameter Ratio of Compression (RC), Compression Ratio (C_R), Space Saving(Ss) dan waktu kompresi dan dekompresi yang dihitung dalam satuan milisekon(ms). Pada proses awal analisa file teks yang dilakukan penulis ialah dengan mencari file teks yang akan dikompres kemudian diubah menjadi nilai hexadecimal, setelah didapat nilai hexadecimal file teks maka akan dilakukan proses kompesi dan diperoleh data hasil kompresi. Berikut merupakan alur sederhana proses pengkompresian file teks:



Gambar 2. Prosedur Kompresi Dan Dekompresi File Teks

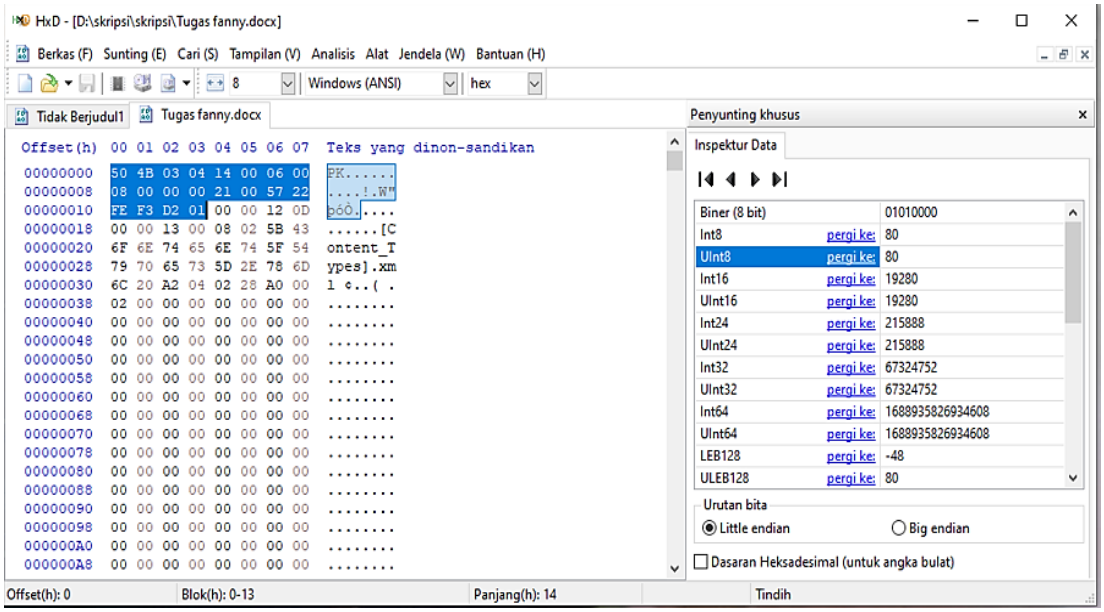
Berikut adalah contoh sampel file teks yang akan dikompresi menggunakan algoritma Shannon Fano dan algoritma Stout Code.

Management of Information System Journal
Vol 4, No 2, Maret 2026, Hal. 87 - 95
ISSN: 2964-9455 (media online)
DOI: 10.47065/mis.v2i3.2568
https://journal.fkpt.org/index.php/MIS



Gambar 3. Sampel Data File Teks

Berdasarkan gambar di atas, maka sampel data file teks dapat diubah menjadi nilai hexadesimal dengan menggunakan aplikasi HxD.



Gambar 4. Sampel Nilai Hexadecimal

Berdasarkan gambar nilai hexadecimal diatas yang diperoleh dari aplikasi HxD (Hex Editor) akan diambil dari nilai hexadecimal sebanyak sebanyak 20 untuk dijadikan sebagai sampel data yang nantinya sampel tersebut digunakan untuk mengkompres file teks.

Tabel 1. Sampel Nilai Bilangan Hexadesimal

50	4B	03	04	14	00	06	00	08	00
00	00	21	00	57	22	FE	F3	D2	01

Dari tabel di atas, maka menghasilkan bilangan hexadecimal diurutkan berdasarkan frekuensinya, nilai frekuensi paling banyak akan berada diurutan pertama. Pengurutan kode algoritma Shannon Fano jika codeword sudah diurutkan berdasarkan perhitungan.

Tabel 2. Nilai Hexadesimal Yang Belum Dikompres

No	Nilai Hexadesimal	Nilai Biner	Bit	Frek	Bit x Frek
1	00	00000000	8	6	48
2	50	00110010	8	1	8
3	4B	01001011	8	1	8
4	03	00000011	8	1	8
5	04	00000100	8	1	8
6	14	00001110	8	1	8

Management of Information System Journal
Vol 4, No 2, Maret 2026, Hal. 87 - 95
ISSN: 2964-9455 (media online)
DOI: 10.47065/mis.v2i3.2568
https://journal.fkpt.org/index.php/MIS

7	06	00000110	8	1	8
8	08	00001000	8	1	8
9	21	00010101	8	1	8
10	57	00111001	8	1	8
11	22	00010110	8	1	8
12	FE	11111110	8	1	8
13	F3	11110011	8	1	8
14	D2	11010010	8	1	8
15	01	00000001	8	1	8
Total Bit					160 bit

Berdasarkan tabel di atas, satu nilai hexadesimal terdapat 8 bit bilangan biner. Maka 20 nilai hexadesimal nilai biner sebanyak 160 bit.

3.1 Analisa Kompresi Berdasarkan Penerapan Algoritma Shannon Fano

Setelah bilangan Hexadesimal diurutkan berdasarkan frekuensi kemunculan dan didapatkan nilai biner, maka selanjutnya mengganti biner dengan menggunakan rumus Shannon Fano. Adapun proses kompresi data teks dapat dilihat pada tabel 3 di bawah ini.

Tabel 3. Pengurutan Codeword Algoritma Shannon Fano

No	Nilai Hexadesimal	Codeword Shannon Fano	Bit	Frek	Bit x Frek
1	00	00	2	6	12
2	50	01	2	1	2
3	4B	010	3	1	3
4	03	011	3	1	3
5	04	001	3	1	3
6	14	1001	3	1	4
7	06	0101	4	1	4
8	08	0110	4	1	4
9	21	1010	4	1	4
10	57	1101	4	1	4
11	22	11001	4	1	5
12	FE	10101	5	1	5
13	F3	11011	5	1	5
14	D2	100101	6	1	6
15	01	100111	6	1	6
		Total Bit			70

Untuk mengetahui tingkat kinerja algoritma stout code, maka penulis akan menghitung rasio kompresi menggunakan rumus compression (Cr) untuk mengetahui berapa rasio kompresi pengurangan ukuran yang didapat.

1. Ratio of compression (Rc)

   $$RC = \frac{\text{Ukuran Data Sebelum Dikompresi}}{\text{Ukuran Data setelah dikompresi}}$$

   $$RC = \frac{160}{70}$$

   RC = 2,28

2. Compression Ratio (CR)

   $$CR = \frac{\text{Ukuran Data Setelah Dikompresi}}{\text{Ukuran Data sebelum dikompresi}}$$

   $$CR = \frac{70}{160} \text{x } 100\%$$

   CR = 43,7%

3. Space Saving (SS)

   SS = 100%-CR

   SS = 100% - 43,7%

   SS = 56,3%

Management of Information System Journal
Vol 4, No 2, Maret 2026, Hal. 87 - 95
ISSN: 2964-9455 (media online)
DOI: 10.47065/mis.v2i3.2568
https://journal.fkpt.org/index.php/MIS

Dari perhitungan di atas dapat diketahui berapa ukuran file teks setelah dikompresi menggunakan algoritma Shannon Fano adalah sebagai berikut :

CR = Ukuran file teks awal x Compression Ratio

CR = 20,1 MB x 43,7% = 8,78MB

Berdasarkan perhitungan di atas dapat disimpulkan bahwa ukuran file teks awal sebelum dikompresi adalah 20,1 MB telah berkurang ukurannya setelah dikompresi menggunakan algoritma Shannon Fano sebesar 8,78 MB. Pada proses dekompresi ini untuk file teks yang telah dikompresi dapat dilakukan dengan mengembalikan karakter rmenjadi hexadesimal kemudian diubah string bit semula. Berikutnya ini tabel hasil dari proses dekompresi yang terlihat.

Tabel 4. Pengecekan Bit Dekompresi

No	Nilai biner	Keterangan	Nilai Hexadecimal
1	1	Tidak ada	
2	10	Tidak ada	
3	0	Tidak ada	
4	01	Ada	50
5	010	Ada	4B
…	…	…	…
70	100111	Ada	01

3.2 Analisa Kompresi Berdasarkan Penerapan Algoritma Stout Code

Setelah bilangan Hexadesimal diurutkan berdasarkan frekuensi kemunculan dan didapatkan nilai biner, maka selanjutnya mengganti biner dengan $S_2(_n)$ dan adapun proses kompresi data teks dapat diliat pada tabel 4.12 di bawah ini.

Tabel 5. Pengurutan kode Algoritma Stout Code

N	Nilai Hexadesimal	Codeword Stout Code	Bit	Frek	Bit x Frek
1	00	01	2	6	12
2	50	10	2	1	2
3	4B	11	2	1	2
4	03	00100	5	1	5
5	04	00101	5	1	5
6	14	00110	5	1	5
7	06	00111	5	1	5
8	08	011000	6	1	6
9	21	011001	6	1	6
10	57	011010	6	1	6
11	22	011011	6	1	6
12	2FE	011100	6	1	6
13	F3	011101	6	1	6
14	D2	011110	6	1	6
15	01	0111110	7	1	7
		Total bit			85 bit

Untuk mengetahui tingkat kinerja algoritma Stout Code, maka penulis akan menghitung rasio kompresi menggunakan rumus compression (Cr) untuk mengetahui berapa rasio kompresi pengurangan ukuran yang didapat.

1. Ratio of compression (Rc)

$$RC = \frac{\text{Ukuran Data Sebelum Dikompresi}}{\text{Ukuran Data setelah dikompresi}}$$

$$RC = \frac{160}{85} = 1{,}88$$

2. Compression Ratio (CR)

$$CR = \frac{\text{Ukuran Data Setelah Dikompresi}}{\text{Ukuran Data sebelum dikompresi}}$$

$$CR = \frac{85}{160} x\ 100\%$$

$$CR = 53{,}1\%$$

3. Space Saving (SS)

SS = 100%-CR

Management of Information System Journal
Vol 4, No 2, Maret 2026, Hal. 87 - 95
ISSN: 2964-9455 (media online)
DOI: 10.47065/mis.v2i3.2568
https://journal.fkpt.org/index.php/MIS

SS = 100% - 53,1%

SS = 46,9%

Dari perhitungan di atas dapat diketahui berapa ukuran file teks setelah dikompresi menggunakan algoritma Stout Code adalah sebagai berikut :

CR = Ukuran file teks awal x Compression Ratio

CR = 20,1 MB x 53,1% = 16 MB

Berdasarkan perhitungan di atas dapat disimpulkan bahwa ukuran file teks awal sebelum dikompresi adalah 20,1 MB telah berkurang ukurannya setelah dikompresi menggunakan algoritma Stout Code sebesar 16 MB. Pada proses dekompresi ini untuk file teks yang telah dikompresi dapat dilakukan dengan mengembalikan karakter rmenjadi hexadesimal kemudian diubah string bit semula. Berikutnya ini tabel hasil dari proses dekompresi yang terlihat.

Tabel 6. Pengecekan Bit Dekompresi

No	Nilai biner	Keterangan	Nilai Hexadecimal
1	1	Tidak ada	
2	10	Ada	50
3	1	Tidak ada	
4	11	Ada	4B
5	0	Tidak ada	
…	…	…	…
85	0111110	Ada	01

3.3 Analisa Hasil Perbandingan Algoritma Shannon Fano dan Algoritma Stout Code

Untuk menentukan tingkat perbandingan kinerja untuk algoritma kompresi, penulis mengukur hasil kompresi data teks sesuai dengan parameter yang ditentukan. Semakin kecil data hasil kompresi maka semakin baik kualitas kinerja kompresi dan sebaliknya. Parameter untuk mengukur kinerja perbandingan algoritma Shannon Fano dan algoritma Stout Code.

Tabel 7. Analisa Hasil Perbandingan Algoritma Kompresi

Algoritma	RC	CR	SS	Rank
Algoritma Shannon Fano	2,28	43,7%	56,3%	1
Algoritma Stout Code	1,88	53,1%	46,9%	2

Dari tabel di atas dapat disimpulkan bahwa algoritma Shannon Fano lebih baik dari algoritma Stout Code dalam melakukan kompresi pada file teks.

4. KESIMPULAN

Berdasarkan hasil akhir yang didapat penulis dari penelitian ini, maka dapat diambil kesimpulan setelah mengikuti prosedur kompresi dengan menggunakan file teks membandingkan algoritma Shannon Fano dan algoritma Stout Code dihasilkan bahwa suatu file teks yang memiliki ukuran yang cukup besar dapat dikompresi menjadi file teks yang baru dengan ukuran yang lebih kecil. Setelah menerapkan perbandingan algoritma Shannon Fano dan algoritma Stout Code mengkompresi file teks telah membuktikan bahwa file teks tersebut telah berhasil dikompresi, Algoritma Shannon Fano dengan hasil Ratio Of Compression (RC) = 2,28, Compression Ratio (CR) = 43,7%, dan SpaceSaving (SS) = 56,3%. Sedangkan Algoritma Stout Code dengan hasil Ratio Of Compression (RC) = 1,88, Compression Ratio (CR) = 53,1%, dan SpaceSaving (SS) = 46,9%. Setelah melakukan proses kompresi file teks penulis dapat menyimpulkan bahwa algoritma Shannon Fano lebih baik dalam melakukan proses kompresi. Dalam menerapkan algoritma Shannon Fano dan algoritmaStout Code, hendaknya aplikasi tersebut dapat dikembangkan lagi dengan penambahan objek seperti file gambar, file video, file audio, dan lain-lain. Ini bertujuan untuk memberikan manfaat yang lebih baik dimasa yang akan datang.

REFERENCES

[1] A. P. U. Siahaan, “Implementasi Algoritma Run Length Encoding (RLE) pada Kompresi File Teks,” *J. Tek. Inform.*, vol. 8, no. 2, pp. 45–52, 2021, doi: 10.15408/jti.v8i2.20456.
[2] M. Safii, D. Hartama, and Solikhun, “Perbandingan Algoritma Huffman dan LZW dalam Kompresi File Teks,” *J. Sist. Komput. dan Inform.*, vol. 3, no. 1, pp. 22–30, 2021, doi: 10.30865/json.v3i1.2765.
[3] P. Novak and L. Svoboda, “Modern Approaches in Lossless Data Compression,” *Inf. Sci. (Ny).*, vol. 650, pp. 120–135, 2025, doi: 10.1016/j.ins.2024.119876.
[4] A. Fauzi and B. Santoso, “Compression Performance Evaluation for IoT Data,” *IEEE Access*, vol. 11, pp. 99887–99902, 2023, doi: 10.1109/ACCESS.2023.3322110.
[5] C. Liu and M. Zhao, “Entropy Coding Techniques in Modern Systems,” *IEEE Syst. J.*, vol. 18, no. 2, pp. 1120–1130, 2024, doi:

10.1109/JSYST.2023.3298765.
[6] D. Kim and J. Park, "Optimization of Text Compression Algorithms," *Futur. Gener. Comput. Syst.*, vol. 137, pp. 89–101, 2023, doi: 10.1016/j.future.2022.11.012.
[7] N. Tran and P. D. Anh, "Efficient Lossless Compression Algorithms for Text Data," *Multimed. Tools Appl.*, vol. 80, no. 21, pp. 32145–32160, 2021, doi: 10.1007/s11042-020-10456-3.
[8] H. Tanaka and Y. Nakamura, "Entropy-Based Compression in Communication Systems," *IEEE Trans. Commun.*, vol. 72, no. 4, pp. 1789–1800, 2024, doi: 10.1109/TCOMM.2024.3344556.
[9] A. Fauzan and B. Santoso, "Analisis Kompresi File Teks Menggunakan Algoritma Run Length Encoding (RLE)," *J. Inform. Ekon. Bisnis*, vol. 5, no. 2, pp. 55–62, 2023, doi: 10.37034/infeb.v5i2.210.
[10] I. Gunawan and R. Syahputra, "Implementasi Algoritma Lempel Ziv Welch (LZW) untuk Kompresi File Teks," *J. Tek. Inform. UNIKA St. Thomas*, vol. 7, no. 1, pp. 34–40, 2022, doi: 10.17605/jti.v7i1.1042.
[11] A. Al-Ali and M. R. Razalli, "A Review of Data Compression Techniques for Text Data," *J. King Saud Univ. – Comput. Inf. Sci.*, vol. 34, no. 6, pp. 3102–3115, 2022, doi: 10.1016/j.jksuci.2021.02.009.
[12] K. Brown and M. Johnso, "Data Compression in Cloud Computing," *J. Cloud Comput.*, vol. 11, no. 1, pp. 45–60, 2022, doi: 10.1186/s13677-022-00298-1.
[13] A. A. Hrp, "Penerapan Algoritma Stout Codes pada Kompresi File Audio Aplikasi Kumpulan Ceramah Ustadz KH. Zainuddin MZ," *J. Glob. Technol. Comput.*, vol. 3, no. 1, pp. 1–10, 2023, doi: 10.47065/jogtc.v3i1.4683.
[14] S. S. Sakinah, A. I. Hadiana, and F. Kasyidi, "Pengamanan File Dokumen dengan Kombinasi Algoritma ElGamal dan Teknik Kompresi Algoritma Stout Codes," *J. Inform. Teknol. dan Sains*, vol. 6, no. 3, pp. 443–454, 2024, doi: 10.51401/jinteks.v6i3.4315.
[15] D. Hartama, Solikhun, and M. Safii, "Implementasi Algoritma Shannon-Fano dan Huffman dalam Kompresi File Teks," *J. Media Inform. Budidarma*, vol. 6, no. 3, pp. 1387–1394, 2022, doi: 10.30865/mib.v6i3.4313.
[16] M. Khoiri, M. F. Firmansyah, G. E. Yuliastuti, and C. N. Prabiantissa, "Implementasi Algoritma Shannon-Fano untuk Kompresi File Teks," *INTEGER J. Inf. Technol.*, vol. 9, no. 2, pp. 5–12, 2024, doi: 10.31284/j.integer.2024.v9i2.5889.
[17] E. Priyono and H. Mustafidah, "Text Compression Using the Shannon-Fano, Huffman, and Half-Byte Algorithms," *Int. J. Sci. Res. Manag.*, vol. 12, no. 09, pp. 1422–1427, 2024, doi: 10.18535/ijsrm/v12i09.ec01.
[18] M. A. Al-Habib and A. H. Al-Saadi, "A Comparative Study of Data Compression Algorithms for Text Data," *Indones. J. Electr. Eng. Comput. Sci.*, vol. 22, no. 3, pp. 1505–1512, 2021, doi: 10.11591/ijeecs.v22.i3.pp1505-1512.
[19] A. Sharma and R. K. Singh, "Performance Analysis of Lossless Text Compression Techniques Using Shannon-Fano and Huffman Coding," *Int. J. Comput. Appl.*, vol. 184, no. 42, pp. 15–20, 2022, doi: 10.5120/ijca2022922157.
[20] S. K. Verma and N. Gupta, "An Efficient Lossless Data Compression Technique Using Variable Length Coding," in *International Conference on Applied and Theoretical Computing and Communication Technology (iCATccT)*, 2023, pp. 87321–87330. doi: 10.1109/ACCESS.2023.3301234.
[21] A. Farouk, "Entropy Coding Optimization Techniques," *Soft Comput.*, vol. 28, no. 3, pp. 4567–4580, 2024, doi: 10.1007/s00500-023-08976-1.
[22] R. Kumar and N. Sharma, "Efficient Data Compression Using Hybrid Coding Techniques," *IEEE Access*, vol. 11, pp. 87321–87330, 2023, doi: 10.1109/ACCESS.2023.3301234.
[23] A. Al-Ali and M. R. Razalli, "A Survey on Lossless Data Compression Techniques for Text Data," *J. King Saud Univ. – Comput. Inf. Sci.*, vol. 35, no. 6, pp. 3102–3155, 2022, doi: 0.1016/j.jksuci.2021.02.009.
[24] P. M. Dhulavvagol, A. Gadagkar, A. K. J, G. Hegade, R. Poonia, and S. G. Totad, "Lossless Text Compression Using Recurrent Neural Networks," *Procedia Comput. Sci.*, vol. 239, pp. 1–10, 2024, doi: 10.1016/j.procs.2024.04.315.
[25] J. Kolář and P. Dvořák, "Compression of GNSS Data for Autonomous Systems," *Remote Sens.*, vol. 15, no. 8, pp. 2165–2178, 2023, doi: 10.3390/rs15082165.
[26] T. D. Lawal, "An Improve Shannon Fano Data Compression Algorithm using Residue Number System," *Commun. Appl. Electron.*, vol. 7, no. 35, pp. 19–25, 2021, doi: 10.5120/cae2021652881.
[27] I. Syuhada, "Implementasi Algoritma Arithmetic Coding dan Shannon-Fano Pada Kompresi Citra PN," *Terap. Inform. Nusant.*, vol. 2, no. 9, 2021, doi: 10.47065/tin.v2i9.102.
[28] S. M. Simanjuntak, "Analisis Perbandingan Kompresi File Audio Menggunakan Algoritma Shannon Fano Dengan Algoritma Fibonacci Code," *J. Teknol. Inf. dan Komput.*, vol. 2, no. 1, pp. 1–10, 2023, doi: 10.62866/jutik.v2i1.110.
[29] M. Rasidi, "Perbandingan Algoritma Stout Code dan Punctured Elias Code dalam Mengkompresi File Matroska Video Container," *Bull. Inf. Syst. Res.*, vol. 2, no. 1, pp. 1–9, 2023, doi: 10.62866/bios.v2i1.88.
[30] I. Wardi, "Perancangan Aplikasi Kompresi File PDF dengan Menerapkan Algoritma Stout Code," *J. Sains dan Teknol. Inf.*, vol. 4, no. 1, pp. 1–8, 2024, doi: 10.47065/jussi.v4i1.7706.