IMPLEMENTASI ALGORITMA RUN LENGTH ENCODING (RLE) UNTUK EFISIENSI PENYIMPANAN DATA TEKS BERBASIS PYTHON

Angelita [1], Triowali Rosandy [2]
[1,2] Teknik Informatika, Institut Informatika dan Bisnis Darmajaya
angelita.2111010026@mail.darmajaya.ac.id

ABSTRAK

Di era digital, kebutuhan akan efisiensi penyimpanan data, terutama file teks berukuran besar, terus meningkat. Penelitian ini mengimplementasikan algoritma Run Length Encoding (RLE) dalam sebuah aplikasi web berbasis Python dan *framework* Flask untuk mengompresi dan mendekompresi file teks. Kami menguji efektivitas algoritma RLE menggunakan model pengembangan *waterfall*, yang meliputi analisis kebutuhan, desain sistem, implementasi, dan pengujian *black box*. Hasil pengujian menunjukkan bahwa algoritma RLE mampu mengurangi ukuran file teks dengan pola berulang hingga 16,67%. Namun, pada data acak tanpa pola, algoritma ini justru meningkatkan ukuran file, menunjukkan adanya keterbatasan. Aplikasi yang dikembangkan ini terbukti efisien dalam menghemat ruang penyimpanan untuk data teks yang memiliki pola berulang.

Kata kunci : *Run Length Encoding*, Kompresi Data, File Teks, Penghematan Data, Python, Optimasi Ukuran File, Algoritma Sederhana.

1. PENDAHULUAN

Di era digital saat ini, manajemen data teks dalam jumlah besar menghadapi tantangan efisiensi penyimpanan. Kapasitas penyimpanan yang terus meningkat seiring bertambahnya data juga berimplikasi pada biaya. Oleh karena itu, kompresi data menjadi solusi utama untuk mengurangi ukuran data tanpa mengorbankan kualitas atau informasi penting[1].

Algoritma Run Length Encoding (RLE) adalah salah satu teknik sederhana yang efektif untuk menghemat ruang penyimpanan data teks dengan mendeteksi dan mengganti pola berulang menjadi representasi yang lebih ringkas[2]. kesenjangan penelitian dalam mengkaji efektivitas dan efisiensi algoritma ini secara spesifik pada aplikasi berbasis web. Kebanyakan penelitian sebelumnya hanya fokus pada kompresi data multimedia atau pengujian sederhana tanpa platform yang spesifik. Oleh karena itu, permasalahan mendasar dalam riset ini adalah bagaimana mengimplementasikan dan menguji secara komprehensif algoritma RLE untuk mengoptimalkan penghematan ruang penyimpanan file teks dalam sebuah aplikasi web.

Penelitian sebelumnya, seperti yang dilakukan oleh Siradjuddin serta Hasan, telah menunjukkan hasil positif dalam kompresi data multimedia dan teks menggunakan RLE dengan Python[1], [3]. Namun, penelitian-penelitian tersebut belum secara spesifik menguji efektivitasnya pada platform aplikasi web. Sebaliknya, penelitian ini secara khusus mengisi kesenjangan tersebut dengan membangun dan mengevaluasi efektivitas RLE dalam optimalisasi penghematan data teks melalui aplikasi berbasis web. Kami juga membandingkan hasilnya dengan aplikasi kompresi lain yang umum digunakan.

Namun, belum banyak penelitian yang secara spesifik mengkaji efektivitas dan efisiensi RLE dalam mengompresi file teks berbasis Python pada platform aplikasi berbasis web. Oleh karena itu, penelitian ini bertujuan untuk mengisi kesenjangan ini dengan mengevaluasi efektivitas RLE dalam optimalisasi penghematan ruang penyimpanan data teks oleh aplikasi berbasis web dan membandingkan hasilnya dengan aplikasi kompresi lain yang umum digunakan.

2. TINJAUAN PUSTAKA

2.1. Kompresi Data

Kompresi data merupakan salah satu cabang ilmu komputer yang berasal dari teori informasi. Teori informasi mempelajari berbagai metode penyimpanan dan pemrosesan pesan, termasuk cara mengurangi redundansi atau informasi tidak berguna dalam data[4]. Semakin banyak redundansi, semakin besar ukuran pesan. Oleh karena itu, tujuan utama dari kompresi data adalah menghilangkan redundansi tersebut untuk mengurangi ukuran file [2].

Teknik kompresi data dibagi menjadi dua kategori utama:

a. Kompresi Lossless – Kompresi yang menjaga integritas data tanpa kehilangan informasi, sehingga data dapat dikembalikan ke bentuk aslinya secara utuh. Cocok digunakan pada file teks, gambar medis, atau gambar satelit. Contoh algoritma lossless: Huffman Coding dan Arithmetic Coding [5].
b. Kompresi Lossy – Kompresi yang menghilangkan sebagian informasi, sehingga hasilnya tidak dapat dikembalikan ke bentuk asli. Cocok untuk file audio, gambar, dan video. Contoh algoritma: Wavelet Compression, Fractal Compression, dan Wyner-Ziv Coding (WZC)[6].

2.2. Python

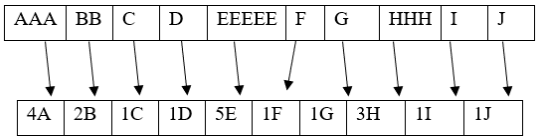
Python adalah bahasa pemrograman serbaguna yang menawarkan berbagai pustaka untuk pengolahan file teks dan multimedia. Python memiliki pustaka seperti `PIL` untuk manipulasi gambar dan `zlib` untuk kompresi data umum [7].

Dalam konteks kompresi menggunakan RLE, Python memberikan fleksibilitas tinggi dalam memodifikasi algoritma sesuai kebutuhan penelitian. Menurut Hasan (2024), Python memudahkan proses manipulasi string, pengembangan logika kompresi, dan pengujian performa[1]. Siradjuddin H, Khairan A, dkk juga menyatakan bahwa Python unggul dalam kecepatan pengembangan dan efisiensi dalam pemrosesan data.

2.3. Algoritma Run Length Encoding (RLE)

Algoritma Run Length Encoding (RLE) menggantikan urutan karakter berulang dalam data dengan pasangan jumlah pengulangan dan karakter[8]. Contoh jika string : "AAAABBCDEEEEEGHHHIJ" dikompresi menggunakan algoritma RLE maka hasilnya menjadi "^4A2^B^1C^1D^5E^1F^1G^3H^1I^ 1J". di mana simbol `^` adalah tanda byte (m), angka setelahnya (n) menunjukkan jumlah pengulangan, dan huruf terakhir (s) adalah karakter yang diulang.

Algoritma RLE ideal untuk data homogen atau berulang, seperti teks atau gambar dengan warna dominan[9]. Untuk hasil yang optimal, pengulangan karakter sebaiknya lebih dari tiga kali. Tanda (m) sebaiknya berupa karakter yang jarang digunakan, seperti `^`, `#`, `~`, atau `|`.



Gambar 1. Proses Decoding

Kelebihan RLE:

a. Sederhana dan mudah diimplementasikan.
b. Efisien untuk data berulang, seperti gambar atau teks log.
c. Termasuk kompresi lossless, sehingga data dapat dikembalikan ke bentuk semula.

Kekurangan RLE:

a. Tidak efektif untuk data acak karena dapat memperbesar ukuran file.
b. Kurang efisien pada data kompleks atau teks dengan variasi karakter tinggi.

2.4. Penelitian Terdahulu

Penelitian oleh Siradjuddin H secara eksplisit mengimplementasikan algoritma RLE untuk kompresi data teks menggunakan Python. Fokusnya adalah pada perancangan dan implementasi aplikasi berbasis web, menunjukkan tren modern dalam penelitian kompresi data yang tidak hanya berfokus pada algoritma inti, tetapi juga pada bagaimana algoritma tersebut diintegrasikan ke dalam platform yang dapat diakses [3]. Hasil penelitian menunjukkan bahwa aplikasi Python yang dirancang berhasil memangkas ukuran data teks, sehingga dapat meminimalkan penggunaan ruang penyimpanan dan memanfaatkannya dengan lebih baik. Kesimpulan ini memvalidasi RLE sebagai metode yang efektif untuk tujuan tersebut.

Penelitian selanjutnya oleh Rahma, S., & Prapanca, yang mengevaluasi kinerja RLE pada berbagai format data, termasuk teks (.txt, .docx, .pdf) dan audio (.mp3, .wav). Temuan mereka menunjukkan bahwa RLE bekerja sangat baik pada format .txt, dengan salah satu sampel data yang memiliki banyak pengulangan karakter mencapai rasio kompresi impresif sebesar -80% [10]. Namun, penelitian ini juga mengidentifikasi batasan algoritma, khususnya kinerja yang tidak optimal pada file .pdf, yang dianggap sebagai batasan masalah dalam studi mereka. Hal ini menguatkan argumen bahwa efisiensi RLE sangat bergantung pada struktur dan karakteristik file yang dikompresi.

Kemudian pada tahun 2018 penelitian oleh Prayoga, E., & Suryaningrum, K. M penelitian ini mengambil pendekatan hibrida dengan mengimplementasikan gabungan algoritma RLE dan Huffman dalam sebuah aplikasi kompresi berbasis web. Tujuan dari pendekatan ini adalah untuk menggabungkan kekuatan kedua algoritma. RLE efektif untuk mengompresi urutan karakter yang berulang, sementara Huffman, yang menggunakan frekuensi kemunculan karakter, lebih optimal untuk data dengan variasi karakter yang tinggi [11]. Temuan dari penelitian ini menunjukkan potensi besar dalam pendekatan kombinasi untuk meningkatkan efisiensi kompresi secara keseluruhan, yang mengindikasikan bahwa RLE tidak harus berdiri sendiri melainkan dapat berfungsi sebagai bagian dari solusi yang lebih besar.

Selanjutnya penelitian oleh Baidoo, P. K. studi ini melakukan perbandingan langsung antara RLE, Huffman, LZW, dan Arithmetic dalam mengompresi data teks. Metodologi pengujian mencakup perbandingan rasio kompresi, kompleksitas file, dan waktu yang dihabiskan untuk kompresi. Penelitian ini menemukan bahwa efisiensi kompresi bervariasi secara signifikan tergantung pada algoritma yang digunakan dan jenis file teks [6]. Kesimpulan utamanya adalah bahwa pengguna harus menentukan tujuan kompresi mereka terlebih dahulu sebelum memilih algoritma yang paling efisien. Penelitian ini memberikan kerangka kerja komparatif yang penting untuk mengevaluasi posisi RLE di antara algoritma kompresi lainnya.

Terakhir penelitian oleh Mansyuri U pada tahun 2021 ini berfokus pada RLE sebagai metode kompresi data teks dan menekankan bahwa RLE sangat sesuai untuk data yang memiliki banyak

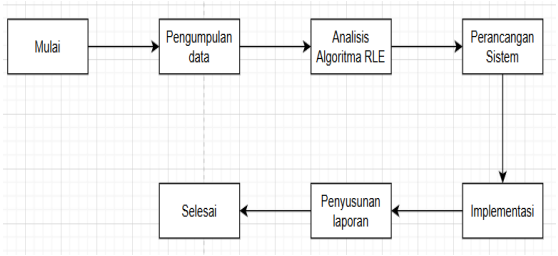
karakter berulang dan berdekatan, seperti contoh "teriakan" yang dibuat secara berlebihan. Studi ini juga secara rinci membahas dua jenis RLE dan memberikan contoh implementasi sederhana[9]. Kesimpulannya memperkuat gagasan bahwa RLE adalah bentuk kompresi yang sederhana tetapi efektif, terutama untuk data yang memiliki pola berulang yang jelas.

3. METODE PENELITIAN

3.1. Alur Penelitian

Alur penelitian dirancang melalui Langkah-langkah berikut:

a. Mengumpulkan data file teks dengan pola acak.
b. Menganalisis algoritma RLE untuk memahami cara kerja kompresi data.
c. Mendesain sistem berbasis Python yang mampu melakukan kompresi dan dekompresi.
d. Melakukan pengujian sistem terhadap berbagai skenario dan ukuran file.
e. Menyusun laporan akhir hasil penelitian



Gambar 2. Alur Penelitian

3.2. Desain Sistem Konsep Algoritma Run Length Encoding (RLE)

Run Length Encoding (RLE) adalah algoritma kompresi lossless yang bekerja dengan cara menggantikan urutan karakter berulang dalam data menjadi pasangan karakter dan jumlah beracak.
Contoh:
Input: AAABBBCCCC
Output: A3B3C4

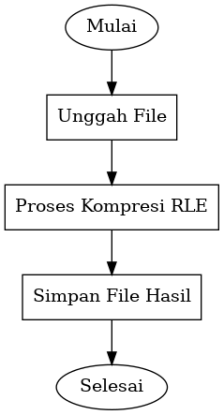
3.3. Implementasi Algoritma RLE dengan Python

Potongan kode atau Pseudocode untuk fungsi encode (kompresi):

```
def encode_rle(data):
    encoded = ""
    i = 0
    while i < len(data):
        count = 1
        char = data[i]
        j = i
        while j < len(data) - 1 and data[j] == data[j+1]:
            count += 1
            j += 1
        encoded += str(count) + char
        i = j + 1
    return encoded
```

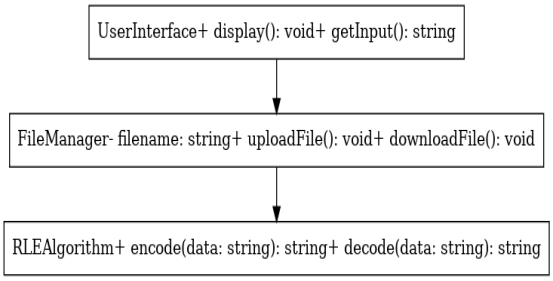
3.4. Perancangan Activity Diagram

Menjelaskan alur aktivitas sistem, mulai dari unggah file hingga proses kompresi dan unduh hasil.



Gambar 3. Diagram Activity

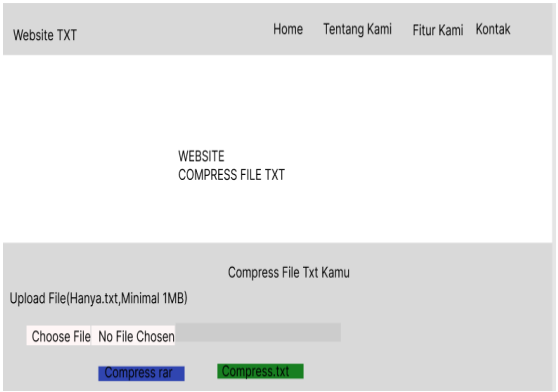
3.5. Perancangan Class Diagram



Gambar 4. Class Diagram

3.6. Perancangan User Interface

Tujuan UI adalah untuk memberikan gambaran tentang aplikasi yang akan dibangun sehingga implementasinya lebih mudah. Untuk pembangunan aplikasi ini, ini adalah rancangan antarmuka pengguna yang akan digunakan.

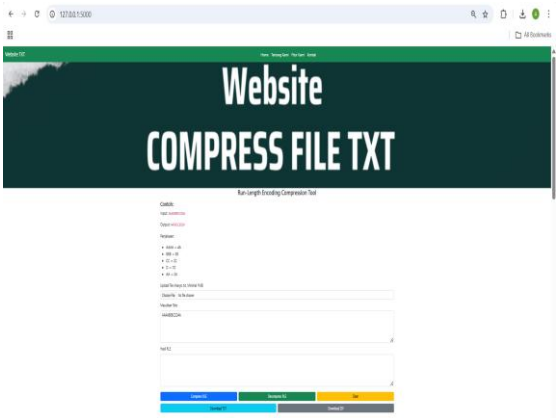


Gambar 5. Perancangan Halaman Home

4. HASIL DAN PEMBAHASAN

Pada hasil penelitian ini menjelaskan hasil dan kemudian menunjukkan penerapan dari program yang sebelumnya telah dirancang.

4.1. Hasil Interface Pengguna



Gambar 6. Interface Halaman Utama

4.2. Pengujian Sistem

Pengujian dilakukan untuk mengevaluasi apakah sistem yang diselidiki, direncanakan, dan diterapkan sesuai dengan ekspektasi. Penguji dalam penelitian ini menggunakan metode pengujian Black Box dan White Box.

4.3. Blackbox Testing

Pengujian sistem merupakan tahapan selanjutnya setelah program atau aplikasi perangkat lunak telah selesai, proses pengujian ini dilakukan dengan metode black-box testing untuk mengevaluasi hasil sistem yang telah dibuat yang mengutamakan pengujian terhadap kebutuhan fungsi dari suatu program dengan menemukan kesalahan fungsi pada perangkat lunak tersebut.

Dalam penelitian ini pengujian sistem menggunakan teknik file berjenis acak merupakan pengujian yang didasarkan pada masukkan file yang akan di kompres untuk memastikan sistem menolak untuk file yang berukuran kurang lebih 1MB.

4.4. Perbandingan Hasil Sebelum dan Sesudah Kompresi

Tabel 1. hasil sebelum dan sesudah di kompresi

No.	Contoh Data (Input)	Ukuran Asli (Karakter)	Ouput RLE	Ukuran Terkompresi (Kareakter)	Efisiensi
1	AAAABBBCCDAA	12	4A3B2C1D2A	10	16,67%
2	ABABCAD	7	1A1B1A1B1C1A1D	14	-100%
3	AAAAAAAAAA	10	10A	3	70%

4.5. Penerapan Algoritma Run Length Encoding(RLE)

Aplikasi kompres file teks ini dikembangkan menggunkan algoritma run length encoding.Berdasarkan hasil penelitian dengan cara mengumpulkan data public yang di ambil dari internet. Dengan jenis file berulang. Studi kasus ini menggunakan dataset teks dengan pola karakter berulang dan acak untuk mengoptimalkan penghematan data file teks berbasis Python. Algoritma Run Length Encoding (RLE) digunakan dalam penelitian ini. Untuk mengetahui seberapa efektif RLE dalam mengurangi ukuran file teks, skenario berikut digunakan.Perhitungan Efisiensi RLE dalam Penghematan Data.

Rumus efisiensi Kompresi:

$$\text{Efisiensi} = \left(\frac{\text{Ukuran Awal} - \text{Ukuran Setelah Kompresi}}{\text{Ukuran Awal}} \right) \times 100\%$$

Cara kerja nya mengubah string kedalam encode namun dapat kembali kan lagi

1. Proses Encoding (Kompresi)
   Misalkan input string:
   AAAABBBCCDAA
   Hasil encoding menggunakan RLE:
   4A3B2C1D2A
   Perhitungan rasio kompresi:
   Ukuran Asli: 12 karakter
   Ukuran terkompresi RLE:10 karakter
   Efisiensi kompres:[((12-10)/12] × 100% =16.67%
2. Proses Encoding (kompresi)
   Jika diberikan string hasil encoding 4A3B2C1D2A, maka hasil encoding nya adalah: I4IAI3IBI2ICI1IDI2IA
   Perhitungan rasio kompresi:
   Ukuran Asli : 10 karakter
   Ukuran terkompresi RLE : 20 karakter
   Efisiensi kompres:[((10-20)/10] ×100% = 100.00%

Jika pola dalam algoritma Run-Length Encoding (RLE) banyak karakter yang berulang secara langsung (bersebelahan), maka hasil encoding RLE tidak akan menghemat banyak ruang menghemat banyak ruang semakin panjang karakter pola encodenya harus panjang juga biar kalau di bagi jadi besar persennya.

5. KESIMPULAN DAN SARAN

Berdasarkan penelitian yang telah dilakukan dalam pengembangan aplikasi kompresi file teks, maka dapat disimpulkan bahwa aplikasi kompresi file teks ini, yang dibangun dengan bahasa

pemrograman Python dan algoritma Run Length Encoding (RLE), terbukti efektif dalam mengoptimalkan proses kompresi file teks dengan pola berulang. Implementasi ini mempermudah pengguna untuk mengurangi ukuran file terkompresi secara efisien. Meskipun demikian, pengembangan lebih lanjut masih diperlukan, seperti penambahan fitur halaman *login* dan perluasan dukungan untuk jenis file lain, misalnya PDF dan gambar, dengan menggunakan metode kompresi yang berbeda dan lebih sesuai.

DAFTAR PUSTAKA

[1] S. Hasan *et al.*, "Penerapan Algoritma Huffman Coding Dalam Menghemat Ruang Penyimpanan Data Multimedia File (Teks dan Gambar) Berbasis Python."

[2] T. Yuspita Hondro, "VIRTUAL : Jurnal Teknik Informatika dan Komputer Penerapan Algoritma Run Length Encoding Untuk Kompresi File MP3", [Online]. Available: https://jurnal.vublish.id/index.php/jurtikom/index

[3] H. K. Siradjuddin, A. Khairan, M. Ridha Albaar, and S. Do Abdullah, "Sistemasi: Jurnal Sistem Informasi Implementasi Algoritma Run Length Encoding (RLE) Pada Kompres Data Teks menggunakan Python Implementation of Run Length Encoding (RLE) Algorithm on Text Data Compress using Python." [Online]. Available: http://sistemasi.ftik.unisi.ac.id

[4] E. Prayoga and K. M. Suryaningrum, "IMPLEMENTASI ALGORITMA HUFFMAN DAN RUN LENGTH ENCODING PADA APLIKASI KOMPRESI BERBASIS WEB," 2018.

[5] M. Robihul Mufid *et al.*, "Image Data Compression in the Public Reporting System in Lamongan using the Huffman Method and Run Length Encoding," 2022. [Online]. Available: https://www.lapor.go.id/instansi/pemerintah-

[6] P. K. Baidoo, "Comparative Analysis of the Compression of Text Data Using Huffman, Arithmetic, Run-Length, and Lempel Ziv Welch Coding Algorithms," *Journal of Advances in Mathematics and Computer Science*, vol. 38, no. 9, pp. 144–156, Aug. 2023, doi: 10.9734/jamcs/2023/v38i91812.

[7] A. Sapto Raharjo, A. Saputra, and S. Yusuf Irianto, "Seminar Nasional Hasil Penelitian dan Pengabdian 2019 IBI DARMAJAYA Bandar Lampung," 2019.

[8] E. R. Roma Hutabarat, "Penerapan Algoritma Run Length Encoding Untuk Kompresi File Database *MySQL," vol. 7, no. 1, 2024, doi: 10.30865/komik.v6i1.8052.

[9] U. Mansyuri, "KOMPRESI DATA TEKS DENGAN METODE RUN LENGTH ENCODING," *Jurnal Ilmiah Sistem Informasi*, vol. 1, no. 2, pp. 102–109, doi: 10.46306/sm.v1i2.

[10] S. Rahma and A. Prapanca, "Analisis Kompresi dan Dekompresi Data Teks dan Audio dengan Algoritma Run Length Encoding (RLE)," *Journal of Informatics and Computer Science*, vol. 02, 2021.

[11] E. Prayoga and K. M. Suryaningrum, "IMPLEMENTASI ALGORITMA HUFFMAN DAN RUN LENGTH ENCODING PADA APLIKASI KOMPRESI BERBASIS WEB," 2018.