SNASTIKOM 2020
Program Studi Teknik Informatika, Fakultas Teknik dan Komputer, Universitas Harapan Medan
website: www.snastikom.com

ANALISIS PERBANDINGAN KOMPRESI DATA TEKS DENGAN MENGGUNAKAN ALGORITMA DIFERENSIASI ASCII DAN HALF-BYTE

Amir Siregar
Universitas Harapan Medan, Jl. HM. Joni No.70 C Medan, satagris@gmail.com

Rosyidah Siregar
Universitas Harapan Medan, Jl. HM. Joni No.70 C Medan, rosyidah_siregar.unhar@harapan.ac.id

Divi Handoko
Universitas Harapan Medan, Jl. HM. Joni No.70 C Medan, divihandoko@gmail.com

Tommy
Universitas Harapan Medan, Jl. HM. Joni No.70 C Medan, tomshirakawa@gmail.com

Andi Marwan Elhanafi
Universitas Harapan Medan, Jl. HM. Joni No.70 C Medan, andimarwanelhanafi@gmail.com

Abstract

Compression data is a method that solidifies necessary information which only needs a small amount of storage space, making it more efficient in saving both time used to compress and transferring information with maximum space left. This method's usage was initially acquired because with time progressing into the modern age, software files have become larger in size, causing information to expand in each data unit. Although there are other methods that can be used, not all of these compression options have the efficiency and competency that may be needed in comparative analysis that an algorithm uses to shrink data. From the variety of methods in compressions, the writer compares two specialized styles in text data compression. The ASCII differential algorithm owns many differences in the process of compression where each ASCII character code is used, whereas the Half-Byte algorithm has similar compression process but only uses the starting 4-bits of each character. Based on the tests and various text data, it can be concluded that the ASCII has a ratio and data size that is overall better than the Half-Byte algorithm.

Keywords:
Data, Comperession, ASCII, Half-Byte

Abstrak

Kompresi data adalah sebuah cara untuk memadatkan data sehingga hanya memerlukan ruang penyimpanan lebih kecil untuk dapat lebih efisien dalam penyimpanan atau mempersingkat waktu pertukaran data dalam memaksimalkan space ataupun storage. Kegunaan metode ini sendiri muncul dikarenakan file yang semakin lama semakin besar sehingga menyebabkan banyak informasi yang harus ditampung oleh sebuah data. Meskipun terdapat banyak metode yang dapat digunakan tidak semua dari metode kompresi memiliki efisiensi serta kompetensi yang dibutuhkan, dalam melakukan analisis perbandingan antara metode dibutuhkan algoritma yang paling baik dalam mengecilkan data. Dari sekian banyak metode kompresi penulis membandingkan dua metode yang dikhususkan dalam kompresi data teks. Algoritama Diferensiasi Ascii memiliki perbedaan dalam proses kompresi dengan menggunakan setiap kode karakter Ascii sedangkan Algoritma Half-byte memiliki persamaan dalam proses kompresi dimana memiliki 4 bit awal yang sama pada semua karakter. Berdasarkan pengujian beberapa data teks dapat disimpulkan algoritma diferensiasi Ascii memiliki rasio dan ukuran data lebih baik dibandingkan dengan algoritma Half-byte.

Kata Kunci:
Data, Kompresi, ASCII, Half-Byte

1. PENDAHULUAN

Kompresi data adalah sebuah cara untuk memadatkan data sehingga hanya memerlukan ruang yang lebih kecil sehingga lebih efisien dalam menyimpannya atau mempersingkat waktu pertukaran data tersebut [1]. Kompresi

merupakan proses pengubahan sekumpulan data menjadi suatu bentuk kode untuk menghemat kebutuhan tempat penyimpanan dan waktu untuk transmisi data [2].

Ada beberapa faktor yang sering menjadi pertimbangan dalam memilih suatu metode kompresi yang tepat, yaitu kecepatan kompresi, sumber daya yang dibutuhkan (memori, kecepatan PC), ukuran file hasil kompresi, besarnya redundansi, dan kompleksitas algoritma. Saat ini terdapat banyak sekali algoritma yang dapat digunakan pada proses kompresi file, beberapa diantaranya algoritma LZW (Lempel-Ziv-Welch), Huffman, Diferensiasi ASCII dan *Half-Byte*.

Algoritma *half-byte* merupakan algoritma yang memanfaatkan empat bit sebelah kiri yang sama secara berurutan terutama pada file-file teks. Saat karakter yang empat bit pertamanya sama diterima secara berderet tujuh kali atau lebih, algoritma ini mengkompres data tersebut dengan bit penanda kemudian karakter pertama dari deretan empat bit yang sama diikuti dengan pasangan empat bit terakhir deretan berikutnya dan ditutup dengan bit penutup Sedangkan algoritma Huffman menggunakan prinsip pengkodean yang mirip dengan kode Morse, yaitu tiap karakter (simbol) dikodekan hanya dengan rangkaian beberapa bit, di mana karakter yang sering muncul dikodekan dengan rangkaian bit yang pendek dan karakter yang jarang muncul dikodekan dengan rangkaian bit yang lebih panjang [3].

Pada metode Diferensiasi ASCII metode ini memanfaatkan nilai selisih dari karakter – karakter ASCII yang terdapat pada sebuah pesan menjadi substitusi yang lebih sederhana dengan menggunakan sebuah window berukuran dinamis untuk memperoleh selisih dari nilai ASCII yang terdapat pada window tersebut sebagai basis dalam menentukan bit substitusi pada file hasil kompresi [4], sedangkan metode *Half-byte* memanfaatkan empat bit sebelah kiri yang sering sama secara berurutan terutama pada file-file teks [5]. *Half Byte* sendiri merupakan metode kompresi yang sudah lama dikenalkan dan telah digunakan pada berbagai penelitian dengan performa yang cukup baik khususnya pada data karakter teks [5][6].

Dalam penelitian ini, penulis menerapkan metode kompresi Diferensiasi ASCII dengan membandingkan metode *Half-Byte* untuk kompresi data dengan format teks. Seperti penelitian yang telah dilakukan sebelumnya, perbandingan dilakukan untuk memperoleh kinerja baik terutama dalam hal ukuran hasil kompresi [7]. Sehingga tujuan penelitian ini adalah untuk mengetahui efesiensi dari masing-masing algoritma yaitu Diferensiasi ASCII dan *Half-Byte* terhadap kompresi data untuk menganalisis kemampuan masing – masing metode terhadap beberapa jenis file terutama file teks yang memiliki pola konten yang berbeda-beda.

1.1 ASCII Difference

Algoritma *Diferensiasi ASCII*memanfaatkan nilai diferensi dari setiap karakter pada tabel ASCII [4]. Nilai deferensi yang cukup rendah dapat digunakan sebagai substitusi dari karakter yang terdapat pada pesan teks. Pada penerapan sederhana metode ini tidak menggunakan *lookup table* atau tabel referensi yang digunakan untuk proses *encoding* dan *decoding*. Proses *encoding* dan *decoding* sepenuhnya menggunakan nilai diferensi dari tiap karakter yang terdapat pada pesan. Namun, tidak tertutup kemungkinan implementasi *lookup table* untuk meningkatkan efektifitas metode pada berbagai jenis berkas digital.

Encoding dilakukan dengan menghitung nilai numeric terendah dan tertinggi dari karakter yang terdapat pada pesan teks menggunakan tabel ASCII. Selanjutnya akan dihitung diferensi dari nilai terendah dan tertinggi tersebut. Nilai diferensi tersebut akan dibagi dua untuk mencari nilai tengah dari nilai diferensi tersebut. Nilai tengah tersebut akan menjadi nilai titik referensi atau *reference point*. Proses *encoding* kemudian dilakukan dengan melakukan substitusi nilai diferensi yang diperoleh dari tiap karakter terhadap titik referensi. Berikut tahapan *encoding* dari differensiasi ASCII :

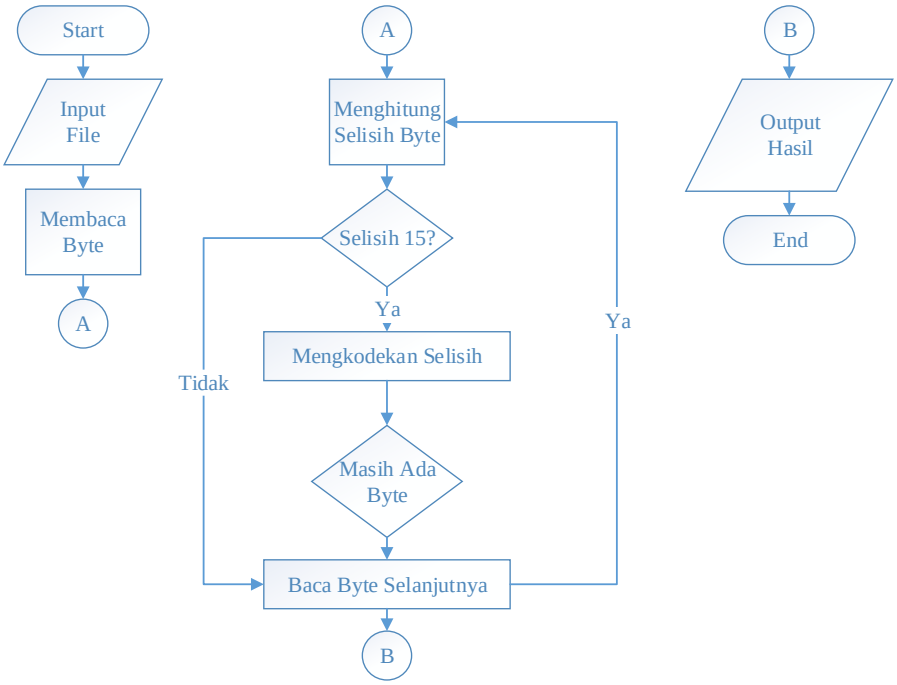
1. Mencari karakter dengan nilai kode ASCII terendah dan tertinggi.
2. Menghitung nilai diferensi dari karakter terendah dan tertinggi tersebut:
   $$d = MaxC - MinC \quad (1)$$
   Dimana :
   d = nilai diferensial.
   MinC = Nilai kode ASCII terendah dari karakter pesan
   MaxC = Nilai kode ASCII tertinggi dari karakter pesan
3. Menghitung nilai tengah dari nilai diferensi dengan menggunakan pembulatan keatas :
   $$Mid = d/2 \quad (2)$$
   Dimana :
   Mid = Nilai tengah
   d/2= diferensial/2
4. Mencari nilai titik referensi :
   p = MinC + Mid (3)

Dimana:
P= titik referensi
MinC=kode ASCII terendah dari karakter pesan
Mid= Nilai tengah

5. Menghitung nilai diferensi tiap karakter terhadap nilai titik referensi.
6. Melakukan *encoding* karakter ditambahkan dengan nilai titik referensi sebagai byte awal.

Decoding dilakukan dengan menghitung nilai ASCII karakter asal menggunakan diferensi antara nilai diferensi karakter yang terkompresi terhadap nilai titik referensi. Berikut tahapan proses *decoding* :

1. Inisialisasi Titik Referensi.
2. Menghitung selisih tiap bit karakter ter-*encode* dengan nilai titik referensi.
3. Konversi nilai diferensi ke karakter ASCII.



Gambar 1. Flowchart Differensiasi ASCII

1.2 Half-Byte

Algoritma *Half-Byte* memanfaatkan empat bit sebelah kiri yang sering sama secara berurutan terutama pada *file-file* teks [6]. Karakter-karakter tersebut memiliki empat bit sebelah kiri yang sama yaitu 0110. Kondisi seperti inilah yang dimanfaatkan oleh Algoritma *Half-Byte*. Saat karakter yang empat bit pertamanya sama diterima secara berderet tujuh kali atau lebih, algoritma ini memampatkan data tersebut dengan bit penanda kemudian karakter pertama dari deretan empat bit yang sama diikuti dengan pasangan empat bit terakhir deretan berikutnya dan ditutup dengan bit penutup.

Algoritma ini paling efektif pada *file* teks dimanabiasanya berisi teks-teks yang memiliki empat *bit* pertama yang sama. Deretan data sebelah kiri merupakan deretan data pada file asli, sedangkan deretan data sebelah kanan merupakan deretan data hasil pemampatan dengan algoritma *Half-Byte*. Langkah-langkah yang dilakukan adalah :

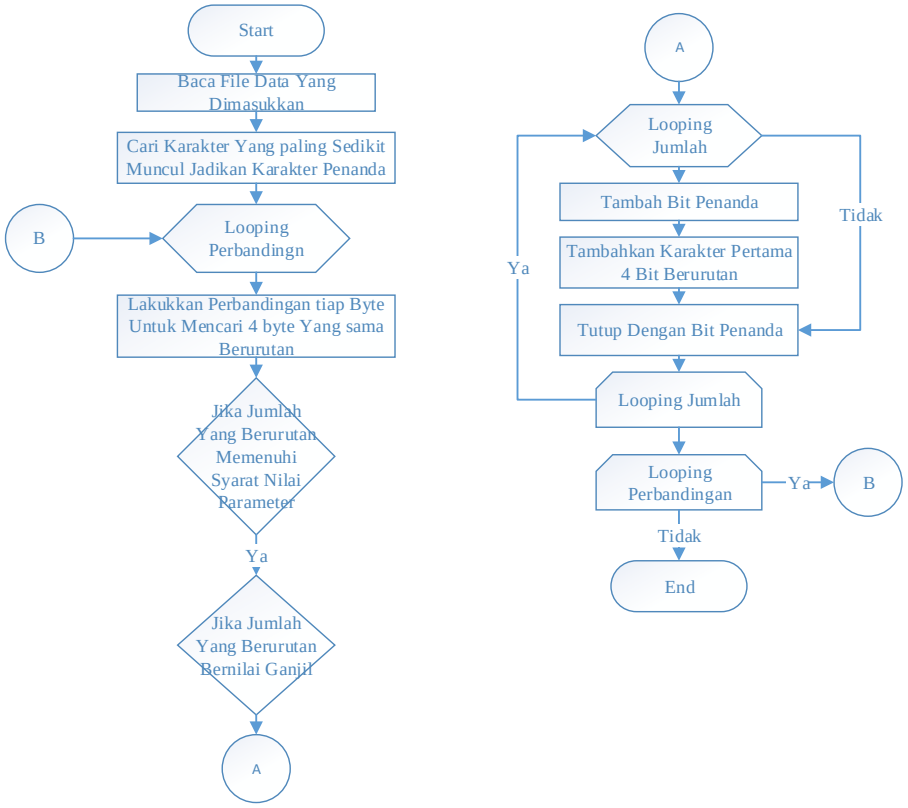
1. Lihat apakah terdapat deretan karakter yang 4 bit permanya sama secara berurutan tujuh karakter atau lebih, jika memenuhi lakukan pemampatan. Pada contoh di atas deretan karakter yang sama secara berurutan sebanyak 9 karakter, jadi dapat dilakukan pemampatan.
2. Berikan bit penanda pada file pemampatan, bit penanda disini berupa 8 deretan bit (1 byte) yang boleh dipilih sembarang asalkan digunakan secara konsisten pada seluruh bit penanda pemampatan. Bit penanda ini berfungsi untuk menandai bahwa karakter selanjutnya adalah karakter pemampatan sehingga tidak membingungkan pada saat mengembalikan file yang sudah dimampatkan ke file aslinya. Pada contoh di atas bit penanda ini dipilih 11111110.
3. Tambahkan karakter pertama 4 bit kiri berurutan dari file asli, pada contoh diatas karakter pertama 4 bit kiri berurutan adalah 01101101.

4. Gabungkan 4 bit kanan karakter kedua dan ketiga kemudian tambahkan ke file pemampatan. Pada contoh di atas karakter kedua dan ketiga adalah 01100101 dan 01101110, gabungan 4 bit kanan kedua karakter tersebut adalah 01011110. Lakukan hal ini sampai akhir deretan karakter dengan 4 bit pertama yang sama.
5. Tutup dengan bit penanda pada file pemampatan.

Untuk melakukan proses pengembalian ke data asli (*decompression*), dilakukan langkah-langkah berikut ini :

1. Lihat karakter pada hasil pemampatan satu-persatu dari awal sampai akhir, jika ditemukan bit penanda, lakukan proses pengembalian.
2. Lihat karakter setelah bit penanda, tambahkan karakter tersebut pada file pengembalian.
3. Lihat karakter berikutnya, jika bukan bit penanda, ambil 4 bit kanannya lalu gabungkan dengan 4 bit kanan karakter di bawahnya. Hasil gabungan tersebut ditambahkan pada file pengembalian. Lakukan sampai ditemukan bit penanda.

Dengan langkah-langkah pengembalian yang telah dijelaskan di atas, akan didapatkan hasil yang sama seperti file aslinya.



Gambar 2. Flowchart Half-Byte

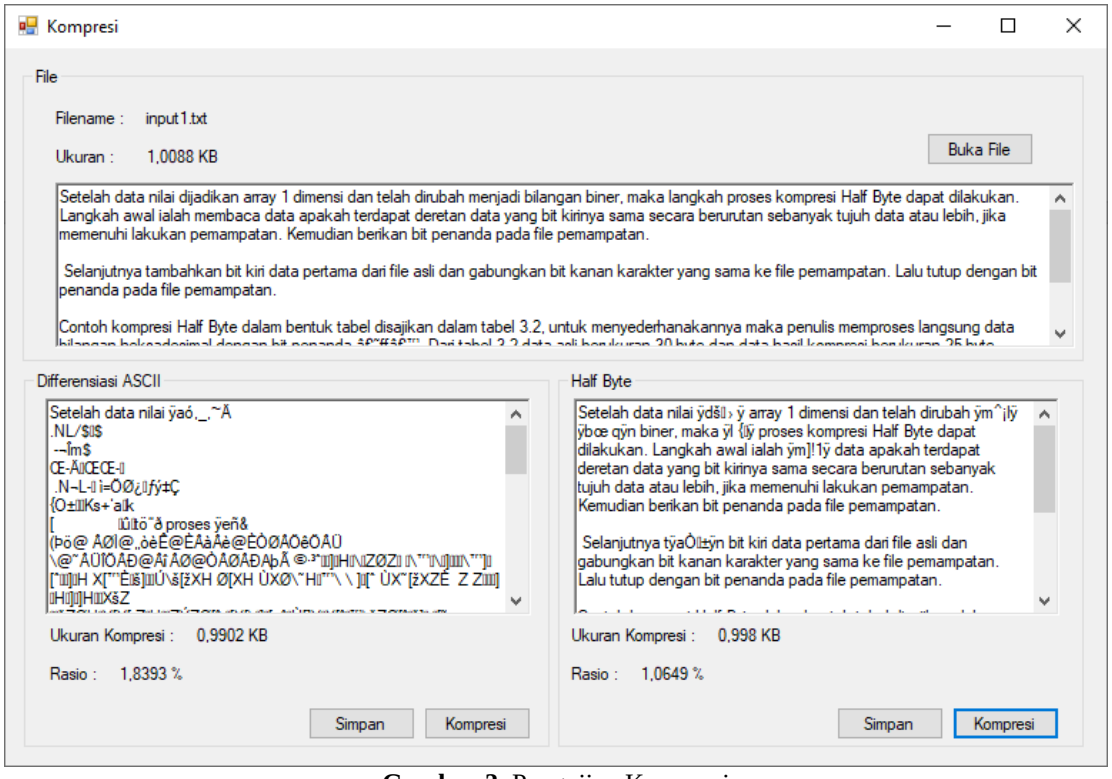
2. HASIL DAN PEMBAHASAN

Pengujian dilakukan dengan melakukan kompresi dan dekompresi pada beberapa *file* teks dan kemudian akan dilanjutkan dengan pembahasan terhadap hasil pengujian yang dilakukan. Pada tabel 1 diatas dapat dilihat file pengujian yang akan digunakan pada pengujian terdiri dari delapan buah file yang memiliki isi dan ukuran yang berbeda – beda. Proses pengujian dilakukan dengan melakukan kompresi dan dekompresi pada file – file diatas secara bergantian.

Tabel 1. File Pengujian

No	Nama File	Ukuran
1	Input1.txt	1.008 Kb
2	Input2.txt	0.222 Kb
3	Input3.txt	13.385 Kb
4	Input4.html	56.781 Kb
5	Input5.jpg	43.027 Kb
6	Contoh.xlxs	7.809 Kb
7	Gambar.bmp	101.123 Kb
8	File_exemple_Wav_1MG.wav	1048.064 Kb

Pada gambar 3 dapat dilihat proses komperesi dari metode diferensiasi ascii dan *half-byte* dimana *file* awal yang berukuran 1.00 kb akan dikomperesi dengan menggunakan kedua metode tersebut sehingga didapatkan hasil komperesi dengan menggunakan metode diferensiasi ascii menghasilkan ukuran komperesi sebesar 0.990 kb dengan rasio 1.83 % sedangkan pada metode *half-byte* menghasilkan ukuran komperesi sebesar 0.998 kb dengan rasio 1.06 %.



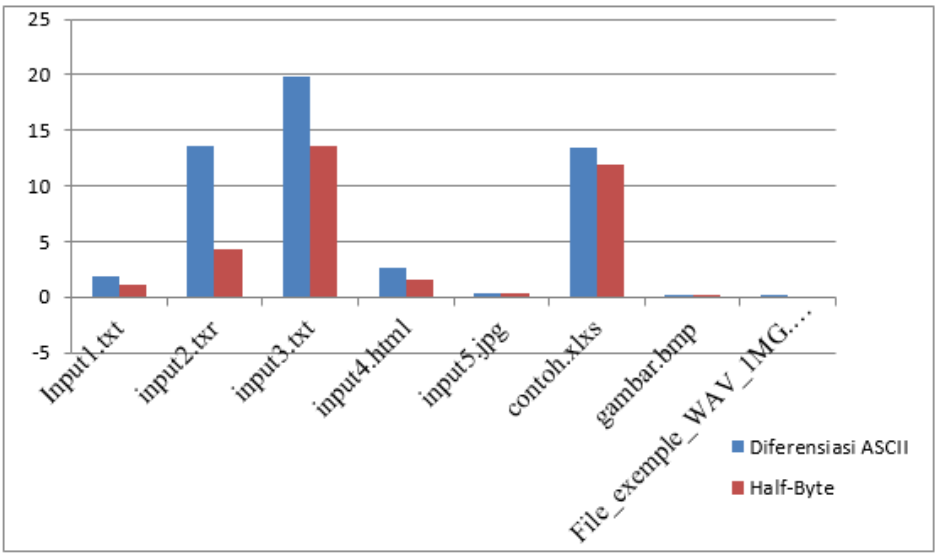
Gambar 3. Pengujian Kompresi

Pengujian yang dilakukan pada penelitian ini bertujuan menganalisa metode kompresi yang digunakan dalam mengecilkan ukuran pesan teks menggunakan metode Diferensiasi Ascii dan *Half-Byte*. Hasil pengujian dapat dilihat pada tabel 2.

Tabel 2. Hasil Rasio Pengujian Black box

No.	Nama File	Ukuran	Differensiasi ASCII		Half Byte	
			Ukuran Kompresi	Rasio Kompresi	Ukuran Kompresi	Rasio Kompresi
1	Input 1.txt	1.008 KB	0.990 KB	1.83 %	0.998 KB	1.06 %
2	Input 2.txt	0.222 KB	0.192 KB	13.59 %	0.212 KB	4.38 %
3	Input3.txt	13.385 KB	10.730 KB	19.83 %	11.574 KB	13.53 %
4	Input4.html	56.781 KB	55.241 KB	2.71 %	55.858 KB	1.62 %
5	Input5.jpg	42.027 KB	42.862 KB	0.38 %	42.875 KB	0.35 %
6	Contoh.xlsx	7.809 KB	6.756 KB	13.48%	6.882 KB	11.86%.
7	Gambar.bmp	101.123 KB	101.104 KB	0.01%	101.118 KB	0.004%.
8	File_exemple_WAV_1MG.wav	1048.064 KB	1047.999 KB	0.006%	1048.157 KB	-0.008%

Berdasarkan hasil pengujian yang dilakukan dapat dilihat bahwa rasio komperesi dari kedua metode yang diteliti memiliki perbedaan walaupun besar perbedaan rasio tidak terlalu signifikan.Pada tabel 4.2 dapat dilihat bahwa rasio terkeci dari metode diferensiasi ascii dari pengujian yang dilakukan adalah sebesar 0.006 % dan paling besar adalah 19.83 % sedangkan pada metode half-byte rasio terkecil yang diperoleh adalah -0.008 % dan rasio komperesi yang terbesar adalah sebesar 13.53 %. Dari perolehan rasio komperesi kedua metode tersebut apat dilihat bahwa metode diferensiasi ascii memberikan rasio komperesi yang lebih baik yang artinya hasil komperesi menggunakan diferensiasi ascii akan memberikan ukuran file yang lebih kecil dibandingkan dengan metode half-byte.



Gambar 4. Grafik Perbandingan Rasio kompresi Diferensiasi ASCII dan Half Byte

Dari Gambar 4 dapat dilihat perbandingan hasil rasio dari kompresi Diferensiasi ASCII dan *Half-Byte*, dimana algoritma Diferensiasi ASCII memiliki rasio yang lebih tinggi disbanding dengan algoritma *Half-Byte*.

3. KESIMPULAN

Berdasarkan pengujian yang telah dilakukan, algoritma Kompresi Diferensiasi Ascii dan *Half-Byte* telah terbukti mampu meng-kompresi data text dengan cukup baik dan dapat diimplementasikan kedalam sistem tanpa adanya kesalahan dalam penggunaan sistem baik pada saat proses kompresi maupun dekompresi data text. analisa hasil kompresi Diferensiasi Ascii dan *Half-Byte* berdasarkan rasio hasil kompresi ukuran data hasil kompresi menunjukkan bahwasanya algoritma Diferensiasi Ascii memiliki efisiensi yang lebih tinggi dibandingkan *Half-Byte* dalam hal kompresi dimana, rasio komperesi terbaik dari metode diferensiasi ascii adalah sebesar 19.83 % sedangkan rasio terbaik dari metode *Half-Byte* sebesar 13.53% untuk file yang sama. Rasio komperesi dari kedua metode sangat bergantung pada kesamaan atau kemiripan *byte* yang terdapat pada file dimana semakin tinggi tingkat kemiripan dari *byte-byte* yang menysun *file* maka akan semakin baik raiso komperesi yang diperoleh,namun diferensiasi ascii memiliki toleransi yang lebih baik terhadap varian byte yang terdapat pada *file* karena berbasi kepada selisih antar *byte* dan ukuran *window* atau blok proses yang lebih dinamis.

DAFTAR PUSTAKA

[1] Jamaluddin, Jamaluddin. (2013). Analisis Perbandingan Kompresi Data dengan Fixed-Length Code, Variable-Length Code dan Algoritma Huffman. 3. 10.17605/OSF.IO/8HJVB.

[2] Mangiri, Herry. (2018). Pembelajaran Kompresi Text dengan Menggunakan Metode Shanon-Fano. Joined Journal (Journal of Informatics Education). 1. 44. 10.31331/joined.v1i1.621.

[3] Satyapratama, Andika, et al. (2015). Analisis Perbandingan Algoritma Lzw Dan Huffman Pada Kompresi File Gambar Bmp Dan Png. Jurnal Teknologi Informasi: Teori, Konsep, dan Implementasi, vol. 6, no. 2, 2015, pp. 69-81.

[4] Tommy, Tommy & Siregar, Rosyidah & Lubis, Imran & E, Andi & Husein, Amir & Harahap, Mawaddah. (2018). A Simple Compression Scheme Based on ASCII Value Differencing. Journal of Physics: Conference Series. 1007. 012022. 10.1088/1742-6596/1007/1/012022.

[5] Supiyandi, S., & Frida, O. (2018). ANALISIS PERBANDINGAN PEMAMPATAN DATA TEKS DENGAN MENGGUNAKAN METODE HUFFMAN DAN HALF–BYTE. ALGORITMA: JURNAL ILMU KOMPUTER DAN INFORMATIKA, 2(1).

[6] Wibowo, Sastya. (2019). PENERAPAN ALGORITMA RUN LENGTH, HALF-BYTE, HUFFMAN UNTUK PEMAMPATAN (COMPRES) FILE. Telematik : Vol 3, No 2, 2011, pp. 19-24

[7] Pujianto, P., Mujito, M., Prasetyo, B. H., & Prabowo, D. (2020). Perbandingan Metode Huffman dan Run Length Encoding Pada Kompresi Dokumen. InfoTekJar: Jurnal Nasional Informatika dan Teknologi Jaringan, 5(1).