# Analisa Kompresi File Teks Menggunakan Algortima Huffman

**Luthfia Sodikin[1,*], Tineke Fatma Putri[1], Taufik Hidayat[2]**

[1,2] Fakultas Teknik, Teknik Komputer, Universitas Wiralodra, Indramayu, Indonesia
Email: [1,*] tinekefatma@gmail.com, [2]thidayat.ft@unwir.ac.id
[*)] luthfiasodikin86903@gmail.com

**Abstrak−** Era Digital saat ini memiliki dampak besar bagi kelestarian alam, selain mengurangi penggunaan kertas dengan adanya berkas berbentuk *file* membuat arsip yang tersimpan lebih tertata rapih dengan kurun waktu yang lama. Di masa pandemi pemanfaatan digital menjadi peran penting dalam kelangsungan kehidupan yang mengharuskan kegiatan tanpa harus kontak langsung. Dengan begitu banyak pekerjaan yang diselesaikan di rumah kemudian dibantu dengan Internet untuk melakukan proses kirim/terima *file. File* merupakan data yang sudah diolah menjadi sebuah informasi berupa teks, video, gambar, dan suara. Dalam proses pengiriman tidak jarang terjadinya kendala yakni ukuran *file* terlalu besar karena media penyimpanan yang digunakan hanya sedikit. Kompresi data adalah ilmu yang menampilkan informasi dalam bentuk yang pendek tujuannya untuk mengurangi jumlah bit yang digunakan untuk menyimpan atau mengirim informasi. Sedangkan *Algoritma Huffman* adalah teknik kompresi yang tidak mengubah informasi data dari aslinya. Prinsip kode *Huffman* yaitu karakter yang paling sering muncul di dalam data dikodekan dengan kode yang jumlah bitnya lebih sedikit, sedangkan karakter yang jarang muncul dikodekan dengan kode yang jumlah bitnya lebih panjang. Kompresi *file* dibutuhkan untuk mempercepat proses pengiriman data antar jaringan komputer. Perkembangan teknologi saat ini menyebabkan kebutuhan data serta perpindahan data dari satu perangkat ke perangkat lain meningkat. Data-data tersebut umumnya dikompresi terlebih dahulu agar proses pertukaran tidak memakan waktu yang lama. Dengan menggunakan *Algoritma huffman* Rasio Kompresi yang dihasilkan untuk *file* audio dan *image* rasio kompresi yang dihasilkan lebih kecil sedangan untuk *file* teks rasio kompresi yang dihasilkan lebih besar. Tingkat keamanan data setelah dikompresi tidak berkurang atau mengalami kerusakan setelah proses kompresi data dilakukan Kecepatan proses kompresi dan dekompresi data setara dengan ukuran dan jenis *file. File* hasil dekompresi akan berhasil seperti *file* semula sebelum dikompresi, kecepatan proses kompresi dan dekompresi data setara dengan ukuran dan jenis *file* dan Kompresi *file* juga kurang berhasil jika isi *file* terlalu sedikit sehingga ukuran *file* asli bisa jadi lebih kecil dari *file* hasil kompresi karena *file* kompresi masih harus menyimpan *huffman tree*-nya.

**Kata Kunci:** File, Data, Huffman, Kompresi, Dekompresi

**Abstract−** The current Digital Era has a big impact on the preservation of nature, in addition to reducing the use of paper with the existence of file-shaped files making the stored archives more neatly organized for a long period of time. In times of pandemic digital utilization becomes an important role in the survival of life that requires activities without having direct contact. With so much work done at home then helped by the Internet to do the process of sending / receiving files. A file is data that has been processed into information in the form of text, video, images, and sounds. In the delivery process, it is not uncommon for the occurrence of constraints, namely the file size is too large because the storage media used is only small. Data compression is the science that displays information in short forms whose purpose is to reduce the number of bits used to store or send information. Huffman's algorithm is a compression technique that does not change the data information from the original. Huffman's code principle is that the characters that appear most often in data are encoded with fewer bits of code, while characters that rarely appear are encoded with code that is longer. File compression is needed to speed up the process of sending data between computer networks. The development of technology today causes the need for data as well as the transfer of data from one device to another increases. The data is generally compressed first so that the exchange process does not take a long time. By using huffman algorithm the resulting compression ratio for audio files and the resulting image compression ratio is smaller while for the resulting text file the resulting compression ratio is greater. The level of data security after compressing is not reduced or damaged after the data compression process is done The speed of the data compression and decompression process is equivalent to the size and type of file. Decompression file will be successful as the original file before it was compressed, the speed of the process of compression and decompression of data equivalent to the size and type of file and File compression is also less successful if the contents of the file are too little so that the original file size can be smaller than the file compression results because the compression file still has to store its huffman tree.

**Keywords**: File, Data, Huffman, Compression, Decompression

# 1. PENDAHULUAN

Saat ini dunia sangat membutuhkan komputer dan jaringan karena pengaruh besar dalam kehidupan kita, dengan menggunakan Internet Keamanan dan perlindungan data telah menjadi isu penting. Ada beberapa skema untuk melakukan enkripsi dan kompresi File.[1] Era Digital saat ini memiliki dampak besar bagi kelestarian alam, selain mengurangi penggunaan kertas dengan adanya berkas berbentuk *file* membuat arsip yang tersimpan lebih tertata rapih dengan kurun waktu yang lama. Masa pandemi merupakan peluang era digital menjadi peran penting dalam kelangsungan kehidupan yang mengharuskan kegiatan tanpa harus kontak langsung. Dengan begitu banyak pekerjaan yang diselesaikan di rumah kemudian dibantu dengan Internet untuk melakukan proses kirim/terima *file*. Proses mengirim informasi secara *real-time* masih mengalami kendala. Salah satunya ialah besarnya jumlah data yang harus dikirim melebihi kecepatan transmisi yang dimiliki oleh perangkat keras yang ada, sehingga *delay time* relatif cukup besar, untuk mentransfer file text dengan ukuran 27 mb dengan lebar *bandwidth* 265 Kbps maka memerlukan waktu lebih dari 10 menit.[2]

File merupakan data yang sudah diolah menjadi sebuah informasi berupa teks, video, gambar, dan suara. Dalam proses pengiriman sering terjadinya kendala yakni ukuran file terlalu besar karena media penyimpanan yang digunakan hanya sedikit. Kompresi data adalah ilmu yang menampilkan informasi dalam bentuk yang pendek tujuannya untuk mengurangi jumlah bit yang digunakan untuk menyimpan atau mengirim informasi. Terdapat 4 (empat) faktor penting dalam kompresi data yakni *Time Process* (waktu yang dibutuhkan dalam menjalankan proses), *Completeness* (kelengkapan data setelah file-file tersebut dikompres), *Ratio Compress* (ukuran data setelah dilakukan kompresi), *Optimaly* (perbandingan apakah ukuran file sebelum dikompres sama atau tidak sama dengan file yang telah dikompres).[3] Kompresi pun dapat dimanfaatkan untuk kebutuhan lain, seperti backup data, proses pengiriman data, serta keamanan data. Pemampatan atau kompresi pada umumnya diterapkan pada mesin komputer, karena setiap simbol yang ditampilkan memiliki bit-bit yang berbeda.[4] Tahapan kompresi digunakan untuk proses pemampatan, dan tahapan dekompresi untuk proses pengembalian file ke bentuk dan ukuran yang semula.

Data yang berukuran besar sering kali sulit untuk disimpan dalam media penyimpanan yang berukuran terbatas. Karena alasan itu, data yang besar sebelum disimpan biasanya dimampatkan terlebih dahulu agar ukurannya lebih kecil dari semula. Salah satu *Algoritma* yang sering digunakan dalam teknik kompresi data adalah kode *huffman*. Teknik ini mampu memampatkan sampai dengan 30% dari ukuran semula. Tetapi proses *decoding string* biner menjadi data kembali masih kurang efisien.[5]

Algoritma Huffman adalah teknik kompresi kompresi yang tidak mengubah informasi data dari aslinya. Algoritma Huffman diperkenalkan oleh David A. Huffman seorang mahasiswa MIT dalam papernya yang berjudul "*A Method for the Construction of Minimum- Redundancy Codes*" dan diterbitkan pada tahun 1952. Prinsip kode Huffman adalah karakter yang paling sering muncul di dalam data dikodekan dengan kode yang jumlah bitnya lebih sedikit, sedangkan karakter yang jarang muncul dikodekan dengan kode yang jumlah bitnya lebih panjang. Algoritma Huffman menggunakan tabel frekuensi kemunculan karakter untuk menggambarkan setiap karakter menjadi kode atau string biner. Kode atau *string* biner yang digunakan untuk mengkodekan setiap karakter dinamakan kode Huffman.[6]

Kode Huffman adalah kode prefiks (*prefix code*). Kode prefiks merupakan himpunan yang berisi sekumpulan kode biner, pada kode prefik ini tidak ada kode biner yang menjadi awal bagi kode biner yang lain. Kode prefiks biasanya direpresentasikan sebagai pohon biner yang diberikan nilai atau label. Untuk cabang kiri pada pohon biner diberi label 0, sedangkan pada cabang kanan pada pohon biner diberi label 1. Rangkaian bit yang terbentuk pada setiap lintasan dari akar ke daun merupakan kode prefiks untuk karakter yang berpadanan. Pohon biner ini biasa disebut pohon Huffman.[7]

Untuk menentukan kode-kode dengan kriteria kode harus unik dan karakter yang sering muncul dibuat kecil jumlah bitnya, kita dapat menggunakan algoritma Huffman untuk melakukan kompresi, dan membentuk Huffman *Tree* (pohon biner Huffman). Adapun langkah-langkah algoritma Huffman sebagai berikut:

1. Langkah pertama adalah menghitung frekuensi
   Kemunculan masing-masing karakter dan mengurutkannya berdasarkan dari frekuensi terkecil kemunculan karakter.
2. Langkah kedua yaitu membuat node masing-masing karakter beserta frekuensinya berdasarkan hasil dari pengurutan sebelumnya.
3. Node-node tersebut diurutkan kembali dari frekuensi terkecil, menjadi node EK digeser ke sebelah kanan node P dan jika menggeset suatu node yang memiliki cabang, maka seluruh node cabangnya diikutkan juga.
4. Setelah pohon Huffman terbentuk, kemudian diberikan tanda bit 0 untuk cabang sebelah kira dan bit 1 untuk setiap cabang sebelah kanan.[8]

Prinsip kode Huffman adalah karakter yang paling sering muncul di dalam data dikodekan dengan kode yang jumlah bitnya lebih sedikit, sedangkan karakter yang jarang muncul dikodekan dengan kode yang jumlah bitnya lebih panjang. Algoritma Huffman menggunakan tabel frekuensi kemunculan karakter untuk

menggambarkan setiap karakter menjadi kode atau string biner. Kode atau *string* biner yang digunakan untuk sebagai cara dalam menganalisa kompresi file teks.

Dari Pembahasan diatas proses pengurangan ukuran file yang dilakukan oleh algoritma Huffman tidak jauh dari pembahasan sistem bilangan, berikut kami jelaskan secara singkat klarifikasi dari bilangan yang digunakan :

a. Bilangan Desimal merupakan bilangan yang direpresentasikan untuk pengguna, dimana pengguna melakukan segala komputasi berdasarkan bilangan desimal. Bilangan desimal berbasis sepuluh menggunakan sepuluh simbol bilangan yaitu: "0", "1", "2", "3", "4", "5", “6", "7", "8" dan "9”

b. Bilangan Biner merupakan bilangan dasar sistem komputer berbasis dua. Oleh karena itu bilangan biner menggunakan 2 simbol bilangan yaitu, "0" dan"1"

c. Bilangan Oktal merupakan bilangan yang digunakan komputer untuk representasi eksternal yang bertujuan membatasi panjang string dan mempermudah pengguna membaca representasi bilangan biner melalui bilangan octal

d. Bilangan Hexadesimal merupakan bilangan yang digunakan komputer untuk representasi eksternal yang bertujuan membatasi panjang string. Bilangan hexadecimal mempermudah pengguna membaca representasi bilangan biner melalui bilangan hexadecimal. Bilangan heksadesimal memiliki basis enam belas dengan simbol bilangan yang digunakan terdiri dari "0”, "1", "2", "3", “4", "5", “6", "7", "8", "9", "A", "B", "C", "D", "E", dan "F”[9]

Proses pengurangan ukuran file yang dilakukan oleh algoritma Huffman menggunakan Sitem Bilangan Biner dan Hexadesimal. Oleh sebab itu penulis memilih algoritma Huffman sebagai cara dalam menganalisa kompresi file teks.

# 2. METODE PENELITIAN

## 2.1. Kompresi Data

Kompresi data (pemampatan data) adalah suatu cara untuk memperkecil jumlah ukuran data dari data aslinya. Dengan Kompresi data diharapkan dapat memperkecil ukuran data dalam ruang penyimpanan.[3] Berbagai jenis kompresi data secara umum digunakan pada komputer pribadi, termasuk kompresi pada program biner,data,suara, dan gambar.[10] Kompresi file dibutuhkan untuk mempercepat proses pengiriman data antar jaringan komputer. Perkembangan teknologi saat ini menyebabkan kebutuhan data serta perpindahan data dari satu perangkat ke perangkat lain meningkat. Data-data tersebut umumnya dikompresi terlebih dahulu agar proses pertukaran tidak memakan waktu yang lama.[9]

Aliran data (*stream*) dapat berupa sebuah file atau *buffer* pada memori. Data dalam konteks kompresi data melingkupi segala bentuk digital dari informasi, yang dapat diproses oleh sebuah program komputer. Bentuk dari informasi tersebut secara luas dapat diklasifikasikan sebagai teks, suara, gambar dan video.[6] Ada banyak teknik kompresi data yang dikategorikan menurut jenis data yang akan dikompresi, yaitu:

- Teknik kompresi untuk citra diam (*still image*) antara lain: JPEG, GIF, dan run length.
- Teknik kompresi untuk citra bergerak (*motion picture*) antara lain: MPEG.
- Teknik kompresi untuk data teks antara lain: half byte
- Teknik kompresi untuk data umum antara lain: LZW, half byte dan huffman.

Metode pemampatan data atau kompresi data dapat dikelompokan dalam dua kelompok besar, yaitu:

1. Metode *lossless*

Lossless data kompresi adalah kelas dari algoritma data kompresi yang memungkinkan data yang asli dapat disusun kembali dari data kompresi. Kompresi data *lossless* digunakan dalam berbagai aplikasi seperti format ZIP dan GZIP. Pada metode kompresi lossless terdapat dua buah model utama yaitu metode *lossless* dengan menggunakan model statistik dan model kamus. Pada model statistik, kompresi data diawali dengan melakukan perhitungan setiap karakter yang ada di dalam file, kemudian dengan statistik karakter yang ada akan dilakukan pengkodean karakter dengan representasi lain. Representasi tersebut apabila dibandingkan dengan karakter asli diharapkan akan lebih kecil ukurannya. Model ini digunakan pada algoritma Huffman dan algoritma Aritmatik. Sedangkan untuk model kamus, kompresi data diawali dengan melakukan perhitungan setiap string yang terdapat di dalam file. *String-string* tersebut akan disusun seperti sebuah indeks dan *string-string* tersebut akan disimbolkan dengan sebuah representasi yang unik. Model ini digunakan pada algoritma Lempel-Ziv dan turunannya (LZW,LZSS, LZRW, dan sebagainya).

2. Metode *lossy*

*Lossy* kompresi adalah suatu metode untuk mengkompresi data dan mendekompresinya, data yang diperoleh mungkin berbeda dari yang aslinya tetapi cukup dekat perbedaaanya. Pada teknik ini akan terjadi kehilangan

sebagian informasi. Data yang telah dimampatkan dengan teknik ini secara umum tidak bisa direkonstruksi sama persis dari data aslinya. Biasanya teknik ini membuang bagian-bagian data yang sebenarnya t idak begitu berguna, tidak begitu dirasakan, tidak begitu dilihat sehingga manusia masih beranggapan bahwa data tersebut masih bisa digunakan walaupun sudah dikompresi. Misalnya pada gambar dan MP3.[11] Lossy kompresi ini paling sering digunakan untuk mengkompres data multimedia (Audio dan gambar statis). Sebaliknya, kompresi *lossless* diperlukan untuk data teks dan file, seperti catatan bank, artikel teks dll.[12]

### 2.2. Dekompresi Data

Dekompresi data adalah pengembalian data seperti semula untuk kemudian dibaca kembali. Dekompresi dibagi menjadi dua, yaitu *lossy* dimana file yang dihasilkan, tidak sama persis seperti sebelum file dikompres, dan lossless dimana file yang dihasilkan akan sama persis seperti file sebelum dikompres.[4] File yang telah berhasil dikompresi akan disimpan menjadi file lain dengan ekstensi yang baru, untuk menggunakannya file harus terlebih dahulu didekompresi dengan menggunakan algoritma Huffman dan mengembalikan data hexadesimal menjadi biner kembali.[9]

### 2.3. Algoritma Huffman

Algoritma Huffman diterdiri dari panjang variabel kode-kode yang disusun dari bit-bit. Simbol dengan probabilitas yang paling tinggi akan memperoleh kode-kode paling pendek sedangkan simbol dengan probabilitas paling rendah akan memproleh kode terpanjang. Kode Huffman memiliki atribut unik yang sempurna, yakni kode-kode yang dapat mengembalikan (*decoder*) kode yang panjang dan tidak menjadi prefik kode Huffman yang lain (decirable).[3] Algoritma Huffman *Encoding* merupakan salah satu algoritma kompresi dengan cara melakukan pengkodean dalam bentuk bit untuk mewakili data karakter. Cara kerjanya:

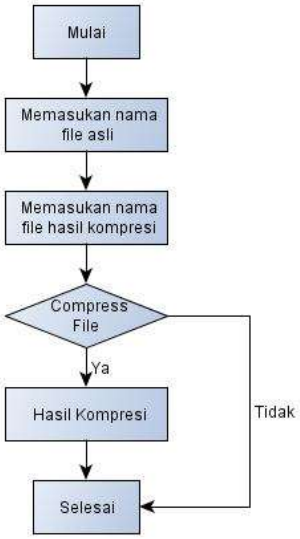
a. Menghitung banyaknya jenis karakter dan jumlah dari masing-masing karakter yang terdapat dalam sebuah file.
b. Menyusun setiap jenis karakter dengan urutan jenis karakter yang jumlahnya paling sedikit ke yang jumlahnya paling banyak.
c. Membuat pohon biner berdasarkan urutan karakter dari yang jumlahnya terkecil ke yang terbesar, dan memberi kode untuk tiap karakter.
d. Mengganti data yang ada dengan kode bit berdasarkan pohon biner.
e. Menyimpan jumlah bit untuk kode bit yang terbesar, jenis karakter yang diurutkan dari frekuensi keluarnya terbesar ke terkecil beserta data yang sudah berubah menjadi kode bit sebagai data hasil kompresi.[9]

Algoritma Huffman bersifat *losseles* artinya setelah dilakukan proses dekompresi tidak ada data yang hilang dari file tersebut.[13] Metode Huffman menggunakan pohon biner untuk menyelesaikan masalahnyamencari kode yang efisien. Pemikiran algoritma ini adalah bahwa setiap karakter *ASCII (American standard for information interchange*) biasanya diwakili oleh 8 bits. [14]

## 3. HASIL DAN PEMBAHASAN

Rasio Kompresi yang dihasilkan untuk file audio dan image rasio kompresi yang dihasilkan lebih kecil sedangan untuk file teks rasio kompresi yang dihasilkan lebih besar. Tingkat keamanan data setelah dikompresi tidak berkurang atau mengalami kerusakan setelah proses kompresi data dilakukan Kecepatan proses kompresi dan dekompresi data setara dengan ukuran dan jenis file.[3] Proses Kompresi data dapat dilihat pada Gambar 1 yakni menjelaskan alur proses saat melakukan pengkompresian. Berikut merupakan penjelasannya:

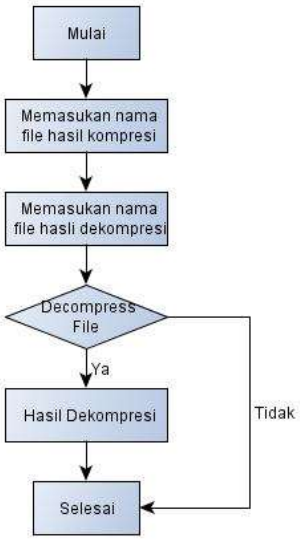
1. User mengunggah file yang telah dipilih,
2. Program membaca data binary dari file dan diubah menjadi data hexadecimal.
3. Data hexadecimal tersebut akan dikompresi menggunakan Huffman Encoding
4. Data disimpan pada server




**Gambar 1.** Proses Kompresi Data

Sedangkan Proses Dekompresi data dapat dilihat pada Gambar 2 yakni menjelaskan alur proses saat melakukan pengdekompresian. Berikut merupakan penjelasannya:

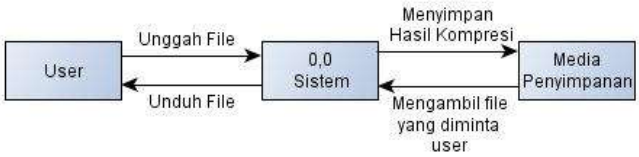
1. User memilih file yang akan didekompres,
2. File akan didekompres menggunakan Huffman Encoding.
3. Data dikembalikan menjadi binary file
4. File siap untuk diunduh.




**Gambar 2.** Proses Dekompresi Data

Data Flow Diagram terbagi menjadi 3 level, yakni level 0, 1 dan 2 berikut ini penjelasan dari masing-masing level:

a. Data Flow Diagram Level 0

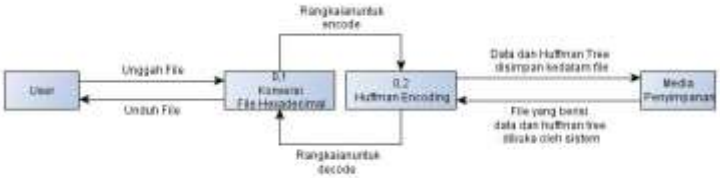



**Gambar 3.** Data Flow Diagram Level 0

Penjelasannya :
User mengunggah file yang akan dikompres ke sistem.

1. Program menyimpan hasil kompresi.
2. User mengirimkan permintaan untuk mengunduh file
3. Program melakukan dekompresi file yang diminta user.
4. File dikirimkan kembali ke pengguna.

b. Data Flow Diagram Level 1




**Gambar 4.** Data Flow Diagram Level 1

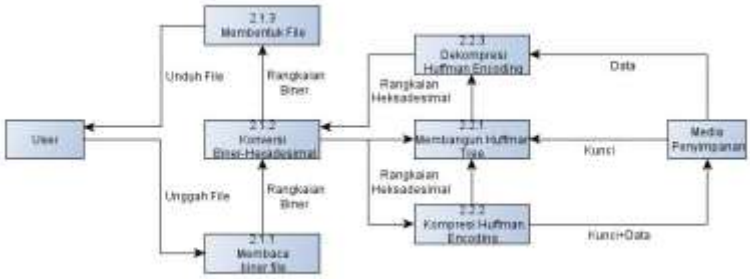
Data flow diagram level 1 menjelaskan proses dalam sistem yang dibagi menjadi 2 proses, yaitu konversi biner hexadecimal dan Huffman Encoding. Berikut merupakan penjelasan proses kompresi:

1. Untuk mengkompresi File yang diunggah, data binary pada file dikonversi menjadi hexadecimal.
2. File dikompresi menggunakan Huffman Encoding.
3. Sistem menyimpan key dan data ke media penyimpanan.

Proses dekompresi dijelaskan sebagai berikut:

1. Untuk mendekompresi File yang sudah tersimpan, dilakukan Huffman Decoding sesuai dengan key dan data yang sudah disimpan.
2. Hasil dekompresi dibangun kembali filenya dari hexadecimal.
3. Hasil Huffman Decoding kemudian dikirimkan ke user.

c. Data Flow Diagram Level 2




**Gambar 5.** Data Flow Diagram Level 2

Pada gambar 5, Data Flow Diagram level 2 digambarkan lebih ringkas. Berikut penjelasan dari saat mengkompresi:

1. Biner file dibaca,
2. Setiap 4 bit dari file, diubah ke dalam bentuk hexadecimal.
3. Dilakukan pembuatan Huffman Tree untuk dijadikan key dari Huffman Encoding yang akan dilakukan dan Huffman Decoding nantinya,Key dan data disimpan menjadi 1 file ke dalam media penyimpanan.

Untuk proses dekompresi, dijelaskan sebagai berikut:

1. File yang sudah tersimpan di media penyimpanan, dipecah menjadi kunci dan data.
2. Data diproses menggunakan Huffman Decoding yang menghasilkan serangkaian hexadecimal.
3. Setelah itu serangkaian tersebut dibuat menjadi file yang kemudian dikirimkan ke pengguna yang memintanya.[9]

Berikut ini hasil perbandingan kompresi dengan dekompresi data pada file teks, image, dan audio :

**Tabel 1.** Hasil Kompresi

| Nama *File* | Ukuran *File* Asli | Ukuran *File* Kompres | Ratio | Waktu (S) |
|---|---|---|---|---|
| T1.txt | 23 kb | 13 kb | 44% | 0.1407 |
| T2.pdf | 449 kb | 428 kb | 5% | 0.6817 |
| T3.doc | 90 kb | 50 kb | 48% | 0.0687 |
| G1.jpg | 1298 kb | 297 kb | 1% | 0.2794 |
| G2.gif | 134 kb | 128 kb | 5% | 0.1309 |
| G3.bmp | 1.787 kb | 322 kb | 83% | 0.7107 |
| A1.mp3 | 2.570 kb | 2.561 kb | 1% | 2.8145 |
| A2.wav | 4.178 kb | 3.473 kb | 17% | 4.0549 |
| A3.wma | 2.905 kb | 2.871 kb | 2% | 3.1551 |

**Tabel 2.** Hasil Dekompresi

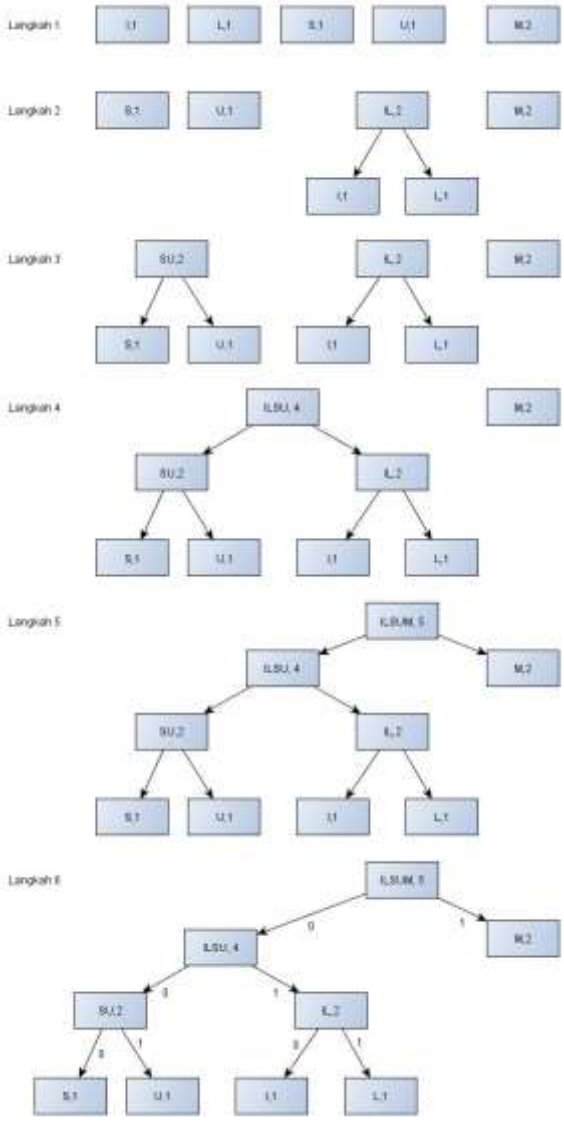
| Nama *File* | Ukuran *File* Kompres | Ukuran *File* Dekompres | Waktu (S) |
|---|---|---|---|
| T1.txt | 13 kb | 23 kb | 0.0031 |
| T2.pdf | 428 kb | 449 kb | 0.3792 |
| T3.doc | 50 kb | 90 kb | 0.1189 |
| G1.jpg | 297 kb | 298 kb | 0.3011 |
| G2.gif | 128 kb | 134 kb | 0.1419 |
| G3.bmp | 322 kb | 1.787 kb | 0.8518 |
| A1.mp3 | 2.561 kb | 2.570 kb | 2.7239 |
| A2.wav | 3.473 kb | 4.178 kb | 4.156 |
| A3.wma | 2.871 kb | 2.905 kb | 3.1423 |

Persentasi kompresi sangat bervariasi, efektivitas kompresi tergantung dari isi file tersebut serta susunan karakter yang terkandung didalam file tersebut, semakin besar file tersebut tetapi variasi susunan karakter didalam file tersebut sedikit maka hasil kompresi akan bisa semakin maksimal, hal sebaliknya apabila file tersebut kecil dan memiliki sedikit karakter sedangkan variasi karakter banyak, maka hasil kompresi akan kurang maksimal, dan apabila dalam file tersebut terdapat sebuah objek yang bukan merupakan sebuah teks maka hasil kompresi juga akan kurang maksimal.[2]

Sebuah file yang akan dimampatkan berisi karakter-karakter “MUSLIM”, huruf yang sama hanya di tuliskan 1 kali, sehingga menjadi M = 4DH = 01001101B, U = 55H = 01010101B, S = 53H = 01010011B, L = 4CH = 01001100B, I = 49H = 01001001B Maka jika diubah dalam rangkaian bit, “MUSLIM” menjadi berukuran 48 bit :

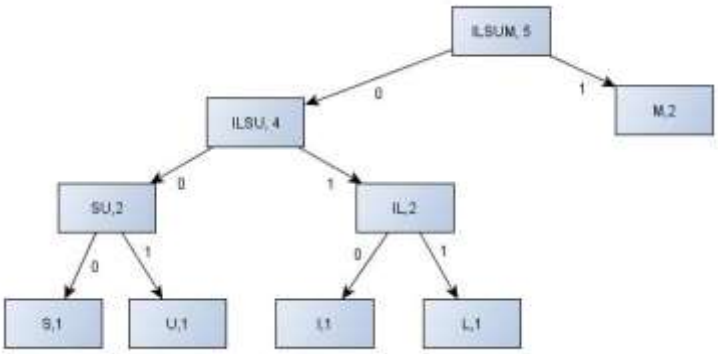
| | | | | | |
|---|---|---|---|---|---|
| 01001101 | 01010101 | 01010011 | 01001100 | 01001001 | 0100110 |
| M | U | S | L | I | M |

Pada string di atas, frekuensi kemunculan M = 2, U = 1, S = 1, L = 1 dan I = 1. Langkah – langkah pembentukan *encoding* pohon Huffman




**Gambar 3.** Langkah Pembentukan *Encoding*

Sehingga hasil dari *encoding* untuk Huffman adalah sebagai berikut :
I = 010, L = 011, S = 000, U = 001, M = 1
Sedangkan Langkah – langkah pembentukan *decoding* pohon Huffman




**Gambar 4.** Langkah Pembentukan *Decoding*[10]

## 4. KESIMPULAN

Kesimpulan yang didapatkan yaitu file hasil dekompresi akan berhasil seperti file semula sebelum dikompresi, kecepatan proses kompresi dan dekompresi data setara dengan ukuran dan jenis file dan Kompresi file juga kurang berhasil jika isi file terlalu sedikit sehingga ukuran file asli bisa jadi lebih kecil dari file hasil kompresi karena file kompresi masih harus menyimpan huffman *tree*-nya. Algoritma Huffman adalah salah satu algoritma kompresi, yang banyak digunakan dalam kompresi teks. Dari ketiga teknik tersebut, proses kompresi data menggunakan Algoritma Huffman memiliki rasio kompresi yang paling tinggi, tetapi memiliki kompleksitas di dalam penyunsunan pohon Huffman. Dari hasil pengujian yang dilakukan, algoritma Huffman dapat mengompres teks sebesar 70% jika dibandingkan dengan menggunakan kode *ASCII* dan sebesar 25,3% jika dibandingkan dengan kita menggunakan 3-bit kode.[12]

Dokumen atau file teks yang sudah dienkripsi menjadi chiperteks memiliki karakter yang lebih banyak atau panjang dibandingkan dengan file teks sebelum dilakukan proses enkripsi. File hasil enkripsi setelah dikompresi memiliki kapasitas setengah kali lebih sedikit dibanding file sebelum dikompresi. File hasil dekompresi akan berhasil seperti file semula sebelum dikompresi File hasil dekripsi sama persis, seperti file awal sebelum dilakukan proses *enkripsi*, misalnya jenis huruf, ukuran huruf dan sebagainya mengalami perubahan setelah dilakukan proses *enkripsi* dan *dekripsi*.[7]

Pada Proses Kompresi File dengan menggunakan kode Huffman akan lebih optimal jika variasi karakter dari informasi tersebut tidak terlalu banyak walaupun frekuensi kemunculannya tinggi karena pohon Huffman yang akan terbentuk tidak terlalu panjang sehingga kode Huffman yang mewakili karakter tersebut menjadi lebih singkat. Dengan begitu berdasarkan pengujian kompresi terhadap file teks dengan karakter tidak berulang, dapat disimpulkan bahwa kompresi dengan metode Huffman jauh lebih efektif.[6]

## UCAPAN TERIMAKASIH

Penulis sangat sangat berterima kasih kepada seluruh pihak terutama kepada Program Studi Teknik Komputer yang telah membantu kami menyelesaikan penulisan ini.

## REFERENCES

[1] K. S. Kasmeera, S. P. James, and K. Sreekumar, "Efficient Compression of Secured Images Using Subservient Data and Huffman Coding," *Procedia Technol.*, vol. 25, no. Raerest, pp. 60–67, 2016, doi: 10.1016/j.protcy.2016.08.081.

[2] A. Pahdi, "Algoritma Huffman Dalam Pemampatan Dan Enksripsi Data," *IJNS - Indones. J. Netw. Secur.*, vol. 6, no. 3, pp. 1–7, 2017, [Online]. Available: http://ijns.org/journal/index.php/ijns/article/view/1461.

[3] A. Wibowo, "Kompresi Data Menggunakan Metode Huffman," *Semantik*, vol. 2, no. 1, pp. 47–51, 2012, [Online]. Available: http://publikasi.dinus.ac.id/index.php/semantik/article/view/70.

[4] E. Prayoga and K. M. Suryaningrum, "Implementasi Algoritma Huffman Dan Run Length Encoding Pada Aplikasi Kompresi Berbasis Web," *J. Ilm. Teknol. Inf. Terap.*, vol. IV, no. 2, pp. 92–101, 2018.

[5] Asrianda, "Kompresi File Menggunakan Algoritma Huffman Kanonik," *TECHSI J. Penelit. Tek. Inform.*, vol. 4, no. 1, pp. 149–160, 2012, [Online]. Available: https://ojs.unimal.ac.id/index.php/techsi/article/view/111.

[6] D. A. Yansyah, "Perbandingan Metode Punctured Elias Code Dan Huffman Pada Kompresi File Text," *J. Ris. Komput.*, vol. 2, no. 6, pp. 33–36, 2015.

[7] K. M. Sudrajat, "Perancangan Aplikasi Pengamanan File Teks Menggunakan Algoritma El Gamal Dan Kompresi File Teks Menggunakan Algoritma Huffman," *Pelita Inform. Inf. dan …*, vol. 8, pp. 173–177, 2019, [Online]. Available: https://ejurnal.stmik-budidarma.ac.id/index.php/pelita/article/view/1814.

[8] Hendrik, "Kombinasi Algoritma Huffman dan Algoritma ROT 13 Dalam Pengamanan File Docx," *J. Inf. Syst. Res.*, vol. 2, no. 1, pp. 40–46, 2020.

[9] K. Geofandy, E. A. Nathaniel, and H. Agung, "Kompresi File Menggunakan Konversi Biner Hexadecimal Dan Algoritma Huffman Encoding," *J. Ilm. Teknol. Infomasi Terap.*, vol. 5, no. 3, pp. 36–46, 2019, doi: 10.33197/jitter.vol5.iss3.2019.295.

[10] I. Algoritma and H. Dan, "Implementasi Algoritma Huffman Dan Lz78 Untuk Kompresi Data," *J. Ris. Komput.*, vol. 3, no. 6, pp. 42–44, 2016, [Online]. Available: https://www.researchgate.net/publication/317671197_IMPLEMENTASI_ALGORITMA_HUFFMAN_DAN_LZ78_UNTUK_KOMPRESI_DATA.

[11] A. F. Siregar, "Perancangan Aplikasi Kompresi File Citra Usg Menggunakan Algoritma Lz78," *J. Pelita Inform.*, vol. 17, no. April, pp. 164–167, 2018.

[12] and T. I. A. Satyapratama, M. Yunus, P. Studi, "ANALISIS PERBANDINGAN ALGORITMA LZW DAN HUFFMAN PADA KOMPRESI FILE GAMBAR BMP DAN PNG," *Ejurnal Stimata*, vol. 6, no. 69–81, p. 2, 2015.

[13] M. R. Irliansyah, S. D. Nasution, and K. Ulfa, "Penerapan Metode Deflate Dan Algoritma Goldbach Codes Dalam Kompresi File Teks," *KOMIK (Konferensi Nas. Teknol. Inf. dan Komputer)*, vol. 1, no. 1, pp. 186–189, 2017.

[14] T. J. Pattiasina, "Analisa Kode Huffman Untuk Kompresi Data Teks," *Teknika*, vol. 1, no. 1, pp. 1–12, 2012, doi: 10.34148/teknika.v1i1.1.