JOURNAL OF COMPUTER SCIENCE AND INFORMATICS ENGINEERING | COSIE
VOL. 03, NO. 4, OKTOBER 2024: 161-169
e-ISSN 2827-8356
p-ISSN 2827-9999

Penerapan Algoritma Huffman Code Untuk Kompresi Dan Dekompresi Pesan

Implementation of Huffman Code Algorithm for Message Compression and Decompression

Akbar Serdano[1], Chairunnisa Azzahra[2], Darus Alpamah[3], Ilham Fajar Batubara[4], Mohd. Wildan Qasthari[5]
[1,2,3,4,5] Universitas Islam Negeri Sumatera Utara
E-mail: [1]akbar.serdano@uinsu.ac.id, [2]nisaazzahra438@gmail.com,
[3]darusalpamah20@gmail.com, [4]ilhambatubara41@gmail.com,
[5]mohdwildanqasthari@gmail.com

Abstrak

Kemajuan teknologi informasi telah membawa perubahan signifikan dalam cara kita mentransmisikan data. Kompresi data adalah metode yang telah lama dikembangkan, yang memungkinkan kita untuk mengurangi ukuran file atau pesan tanpa kehilangan data penting pada file tersebut. Dengan menggunakan algoritma Huffman Code ukuran pesan di kompresi dapat menghemat atau mengurangi ukuran kurang lebih 70%. Huffman merupakan algoritma kompresi lossless, maka pada saat dekompresi akan menghasilkan data yang identik dengan data asli sebelum dikompresi. Algoritma Huffman Code diimplementasikan ke dalam program untuk menguji data yang digunakan. Data yang digunakan untuk penelitian ini berasal dari teks pesan singkat.

Kata kunci: Dekompresi, Huffman, Kompresi

Abstract

Advances in information technology have brought significant changes in the way we transmit data. Data compression is a long-developed method that allows us to reduce the size of a file or message without losing important data in the file. By using the Huffman Code algorithm the size of the message in compression can save or reduce the size by approximately 70%. Huffman is a lossless compression algorithm, so when decompressed it will produce data that is identical to the original data before compression. The Huffman Code algorithm is implemented into the program to test the data used. The data used for this research comes from short message text.

Keywords: Compression, Decompression, Huffman

1. PENDAHULUAN

Kemajuan teknologi informasi telah membawa perubahan signifikan dalam cara kita berinteraksi, menyimpan, dan mentransmisikan data. Dalam era di mana volume data terus meningkat, efisiensi penyimpanan dan transmisi data sangat penting. Kompresi data adalah solusi penting yang telah lama dikembangkan, yang memungkinkan kita untuk mengurangi ukuran file atau pesan tanpa kehilangan data penting pada file tersebut.

Kompresi adalah proses mengubah sekumpulan data menjadi kode untuk menghemat ruang penyimpanan dan waktu transmisi data.[1] Kompresi merupakan proses pengkodean informasi

e-ISSN 2827-8356
p-ISSN 2827-9999

dengan menggunakan bit atau unit pembawa informasi yang lebih kecil untuk mewakili data yang tidak dikodekan dengan sistem pengkodean tertentu, sehingga data tersebut dikompresi menjadi lebih kecil untuk mengurangi waktu transmisi data.[2] Data yang baru dibuat setelah proses kompresi kemudian dikembalikan ke data awal melalui proses yang dikenal sebagai dekompresi.[3] Dekompresi merupakan proses mengembalikan data yang dikompresi menjadi data asli.[4] Proses kompresi dan dekompresi dilakukan menggunakan metode Huffman code.

David A. Huffman memperkenalkan algoritma Huffman sebagai metode kompresi data yang tidak mengubah informasi data asli.[5] Algoritma Huffman merupakan algoritma yang dapat digunakan untuk mengkompres data sehingga menghasilkan ukuran file yang lebih kecil dari ukuran file aslinya, dan dapat menghemat ruang penyimpanan.[6]

Penelitian terdahulu dalam proses kompresi file, algoritma huffman code dan kode unary hampir sama. Setelah tahap dekompresi, ukuran file yang telah dikompresi dapat kembali ke keadaan semula dan dapat dibuka kembali sesuai dengan data file aslinya.[7] Dalam penelitian lain, penerapan metode Huffman untuk kompresi dan dekompresi citra digital dapat menghasilkan sebuah file yang lebih menghemat ruang penyimpanan.[3]

Dalam penelitian sebelumnya tentang analisa kompresi file teks menggunakan Algortima Huffman, kecepatan proses kompresi dan dekompresi data sebanding dengan jenis dan ukuran file.[8] Proses kompresi data menggunakan algoritma ini adalah yang memiliki rasio kompresi tertinggi. Hasil pengujian menunjukkan bahwa algoritma Huffman dapat mengompres teks sebesar 70%.[5] Penerapan sebelumnya tentang huffman code menjelaskan jika file dokumen berbasis teks berukuran besar yang hanya terdiri dari susunan karakter, maka proses kompresi yang dilakukan oleh aplikasi yang dibuat akan sangat efektif.[6]

Pada penelitian ini penulis ingin mengembangkan penelitian terdahulu dengan melakukan kompresi dan dekompresi pesan dengan judul penelitian "Penerapan Algoritma Huffman untuk Kompresi Dan Dekompresi Pesan". Penulis merasa algoritma Huffman relevan digunakan dalam mengompresi pesan atau teks, dari segi kecepatan dan ukuran hasil kompresi. Serta dapat mengembalikan data yang sudah di kompresi kebentuk awal.

2. METODE PENELITIAN

1. Kompresi Data

Kompresi adalah metode pemampatan data yang menghasilkan data dengan ukuran yang lebih kecil daripada ukuran aslinya. Kompresi bekerja dengan mencari pola berulang pada data dan menggantinya dengan penanda tertentu.[3] Kompresi data adalah proses mengkodekan informasi dalam rangkaian bit yang lebih pendek daripada rangkaian bit yang diperlukan dari data sebelum dikompresi dengan metode pengkodean tertentu.[1] Lossless Compresion adalah jenis algoritma kompresi data yang memungkinkan data asli direkonstruksi dari data terkompresi. Lossy Compression merupakan salah satu metode kompresi, data yang dihasilkan bisa saja berbeda dengan aslinya namun tingkat perbedaannya cukup dekat.[5]

2. Dekompresi Data

Proses mengembalikan data ke kondisi awalnya agar dapat dibaca kembali disebut dekompresi data.[2] Sama seperti kompresi, dekompresi terbagi menjadi dua, yaitu dekompresi menghasilkan data yang identik dengan data asli sebelum dikompresi. Oleh karena itu, metode tersebut dinamakan kompresi lossless. Sebaliknya jika hasil dekompresi menghasilkan data yang tidak sepenuhnya sama dengan data asli sebelum didekompresi, karena ada beberapa data yang terhapus karena dianggap tidak penting namun tidak mengubah informasi yang terkandung di

e-ISSN 2827-8356
p-ISSN 2827-9999

dalamnya. maka metode tersebut dinamakan kompresi lossy.[3] Untuk mengembalikan data ke bentuk awalnya, proses dekompresi dilakukan pada data setelah dilakukan proses kompresi.

3. Huffman Code

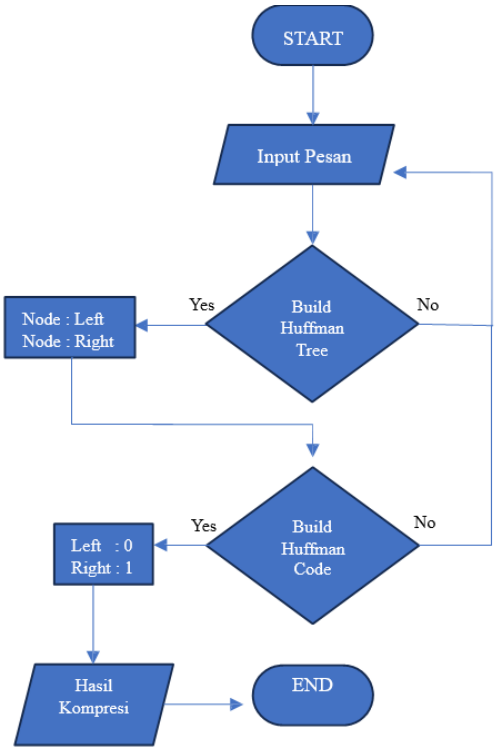
Huffman adalah algoritma kompresi lossless yang banyak digunakan dalam program kompresi karena metode ini relevan untuk mengkompresi teks atau file program.[3] Frekuensi kemunculan data menentukan algoritma Huffman. Data yang sering muncul akan dikodekan dengan jumlah bit yang sedikit, tetapi data yang tidak sering muncul akan dikodekan dengan gabungan bit yang panjang.[4]

3. HASIL DAN PEMBAHASAN

Algoritma Kompresi Huffman Code:

1. Pertama-tama, kita perlu menganalisis frekuensi kemunculan setiap karakter dalam data. Karakter yang sering muncul akan memiliki kode yang lebih pendek. Karakter yang jarang muncul akan memiliki kode yang lebih panjang.
2. Setelah menganalisis frekuensi, kita akan membentuk pohon Huffman. Pohon ini terdiri dari simpul-simpul yang mewakili karakter dan frekuensi kemunculannya. Karakter dengan frekuensi terendah berada di bagian bawah pohon, sedangkan karakter dengan frekuensi tertinggi berada di bagian atas.
3. Setelah membentuk pohon Huffman, kita dapat menentukan kode Huffman untuk setiap karakter. Karakter yang berada di sebelah kiri akan memiliki kode 0, sedangkan karakter yang berada di sebelah kanan akan memiliki kode 1.
4. Setelah menentukan kode Huffman, kita dapat mengompresi data dengan menggantikan setiap karakter dengan kode Huffman yang sudah ditentukan.

e-ISSN 2827-8356
p-ISSN 2827-9999

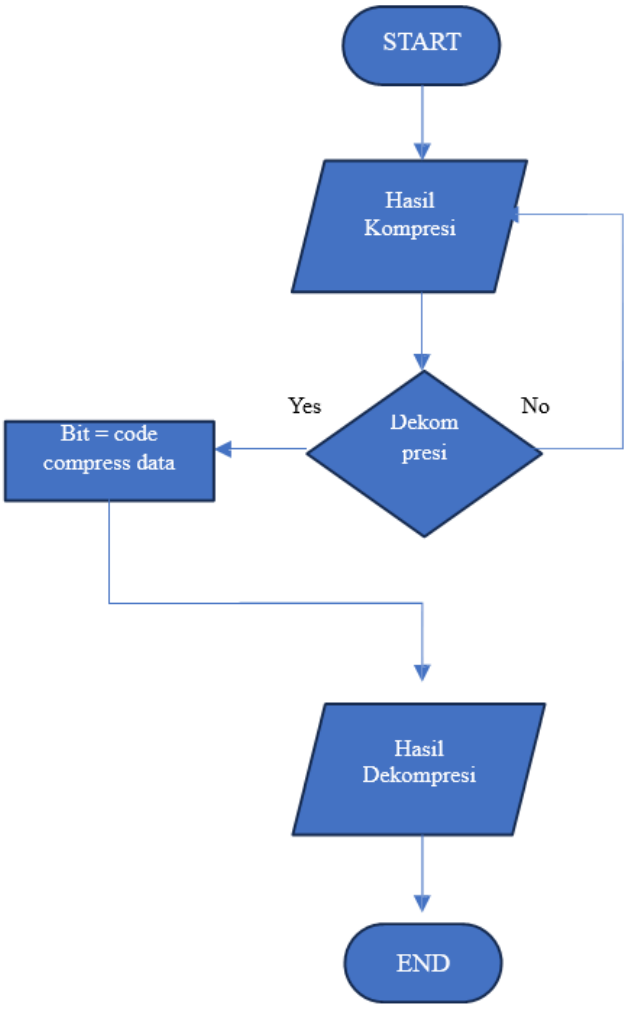


Gambar 1. Flowchat Kompresi

Algoritma Dekompresi Huffman Code:

1. Pertama-tama membaca data yang dihasilkan dari proses kompresi.
2. Kemudian membangun pohon Huffman dengan membaca informasi tambahan yang disertakan dalam data terkompresi, seperti struktur pohon atau tabel frekuensi simbol.
3. Baca satu bit data terkompresi dari akar pohon Huffman.
4. Perhatikan urutan bit yang dibaca saat melakukan traversing pohon Huffman.
5. Setelah mencapai daun pohon, tafsirkan simbol dan masukkan ke dalam pesan terdekompresi.
6. Kemudian kembali ke akar pohon dan ulangi prosedur ini untuk setiap bit berikutnya dalam data terkompresi sampai pesan terdekompresi yang lengkap dibuat.

e-ISSN 2827-8356
p-ISSN 2827-9999



Gambar 2. Flowchat Dekompresi

Contoh:

Melakukan kompresi pada teks “malam selasa” dengan metode huffman code, malam selasa = 12 x 8 = 96
011011010110000101101100011000010110110100100000011100110110010101101100011000010111001101100001

e-ISSN 2827-8356
p-ISSN 2827-9999

ASCII	Biner	ASCII	Biner
A	01000001	O	01001111
B	01000010	P	01010000
C	01000011	Q	01010001
D	01000100	R	01010010
E	01000101	S	01010011
F	01000110	T	01010100
G	01000111	U	01010101
H	01001000	V	01010110
I	01001001	W	01010111
J	01001010	X	01011000
K	01001011	Y	01011001
L	01001100	Z	01011010
M	01001101	space	00100000
N	01001110		

ASCII	Biner	ASCII	Biner
a	01100001	n	01101110
b	01100010	o	01101111
c	01100011	p	01110000
d	01100100	q	01110001
e	01100101	r	01110010
f	01100110	s	01110011
g	01100111	t	01110100
h	01101000	u	01110101
i	01101001	v	01110110
j	01101010	w	01110111
k	01101011	x	01111000
l	01101100	y	01111001
m	01101101	z	01111010

Gambar 3. Tabel Ascii Huruf A-Z

Langkah Pertama:
Kita perlu menganalisis frekuensi kemunculan setiap karakter dalam data kita. Karakter yang sering muncul akan memiliki kode yang lebih pendek. Karakter yang jarang muncul akan memiliki kode yang lebih panjang.

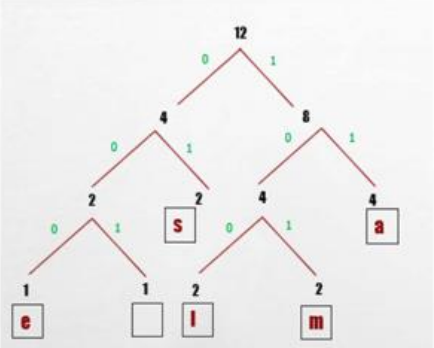
Nilai	Frekuensi
a	4
m	2
l	2
s	2
e	1
spase	1
Total	12

Gambar 4. Frekuensi Setiap Karakter

Langkah berikutnya:
Setelah menganalisis frekuensi, kita akan membentuk pohon Huffman. Pohon ini terdiri dari simpul-

e-ISSN 2827-8356
p-ISSN 2827-9999

simpul yang mewakili karakter dan frekuensi kemunculannya. Karakter dengan frekuensi terendah berada di bagian bawah pohon, sedangkan karakter denganfrekuensi tertinggi berada di bagian atas.



Gambar 5. Pohon Huffman

Langkah selanjutnya:
Setelah memiliki pohon Huffman, kita dapat menentukan kode Huffman untuk setiap karakter. Kode Huffman adalah urutan kode biner yang diperoleh dengan melacak jalur dari akar pohon ke setiap karakter. Karakter yang berada di sebelah kiri akan memiliki kode 0, sedangkan karakter yang berada di sebelah kanan akan memiliki kode 1.

Nilai	Kode Huffman
e	000
	001
s	01
l	100
m	101
a	11

Gambar 6. Kode Huffman

Setelah menentukan kode Huffman, kita dapat mengompresi data dengan menggantikan setiap karakter dengan kode Huffman yang sesuai. Kode Huffman yang lebih pendek akan menghasilkan kompresi yang lebih efisien.

Berikut merupakan contoh program yang dapat dilihat pada gambar 6:

e-ISSN 2827-8356
p-ISSN 2827-9999

```
Masukkan kata atau kalimat: malam selasa

Huffman Tree:
Root: None - 12
    Left: None - 4
        Left: None - 2
            Left: e - 1
            Right:   - 1
        Right: s - 2
    Right: None - 8
        Left: None - 4
            Left: l - 2
            Right: m - 2
        Right: a - 4

Huffman Codes:
e: 000
 : 001
s: 01
l: 100
m: 101
a: 11

Compressed Data:
10111100111010010100010011 0111

Decompressed Data:
malam selasa
```

Gambar 6. Hasil Program Huffman

Melakukan kompresi pada pesan “malam selasa” dengan algoritma Huffman Code, malam selasa = 12 x 8 = 96 bit. Hasil kompresi menggunakan algoritma Huffman Code dari pesan malam selasa = 10111100111000101000100110111. Maka ukuran dari pesan malam selasa menjadi 29 bit. Kemudian proses dekompresi dilakuan dengan membaca data yang dihasilkan dari proses kompresi data.

Tabel 1 Hasil Dekompresi data

101	11	100	11	101	00	01	000	100	11	01	11
m	a	l	a	m		s	e	l	a	s	a

Algorithm Huffman Code memiliki beberapa keunggulan dibandingkan algoritma kompresi lainnya, terutama dalam hal implementasi yang relatif mudah dan kompresi tanpa kehilangan.[9] Untuk data dengan frekuensi karakter yang tidak merata, Huffman Code sangat efektif karena kode biner yang lebih pendek diberikan kepada karakter yang lebih sering muncul, yang menghasilkan kompresi terbaik. Kecepatan dan fleksibilitas Huffman lebih baik tanpa memerlukan pola data tertentu. Ini berbeda dengan algoritma seperti Encoding Run-Length (RLE), yang hanya berfungsi untuk data dengan pola berulang, atau Lempel-Ziv-Welch (LZW), yang lebih kompleks dan membutuhkan kamus. Selain itu, Huffman lebih mudah digunakan dan lebih banyak digunakan dalam aplikasi kompresi seperti format ZIP dan PNG. Huffman menjadi pilihan utama untuk banyak aplikasi yang membutuhkan kompresi efisien dan sederhana.[10]

e-ISSN 2827-8356
p-ISSN 2827-9999

4. KESIMPULAN DAN SARAN

Dengan menggunakan algoritma Huffman Code ukuran pesan di kompresi menjadi 29 bit yang sebelumnya berukuran 96 bit, maka 29 / 96 x 100% = 30,20%, sehingga kita dapat menghemat atau mengurangi ukuran kurang lebih 70%. Huffman merupakan algoritma kompresi lossless, maka pada saat dekompresi akan menghasilkan data yang identik dengan data asli sebelum dikompresi. Pengembangan berikutnya dapat melakukan pengujian yang lebih luas terhadap berbagai jenis data, bukan hanya teks pesan singkat, tetapi juga gambar, audio, atau video, untuk mengetahui seberapa efektif algoritma Huffman Code dalam berbagai situasi. Selain itu, dapat dipertimbangkan untuk menggunakan kombinasi algoritma kompresi lain, seperti coding aritmetik atau LZW, untuk mencapai kompresi yang lebih efisien pada data dengan karakteristik yang berbeda. Dan juga menggunakan dataset yang lebih besar dan variatif untuk mendapatkan hasil yang lebih representatif. Hal ini juga dilakukan untuk menentukan sejauh mana algoritma Huffman dapat dioptimalkan dalam aplikasi.

DAFTAR PUSTAKA

[1] T. J. Pattiasina, "ANALISA KODE HUFFMAN UNTUK KOMPRESI DATA TEKS," TEKNIKA, vol. 1, no. 1, pp. 1–12, 2012, [Online]. Available: https://ejournal.ikado.ac.id/index.php/teknika/article/view/1/

[2] K. M. Prayoga, Eka; Suryaningrum, "IMPLEMENTASI ALGORITMA HUFFMAN DAN RUN LENGTH ENCODING PADA APLIKASI KOMPRESI BERBASIS WEB," J. Ilm. Teknol. Inf. Terap., vol. 4, no. 2, pp. 92–101, 2018.

[3] K. K. Mahesa, "Rancang Bangun Aplikasi Kompresi Dan Dekompresi Pada Citra Digital Menggunakan Metode Huffman," PROCESSOR, vol. 12, no. 1, pp. 997–1012, 2017.

[4] H. Dewi, Kuni Yustika; Nurwasito, "Implementasi Algoritma Huffman untuk Kompresi Gambar pada Jaringan 6LoWPAN," J. Pengemb. Teknol. Inf. dan Ilmu Komput., vol. 6, no. 4, pp. 2008–2017, 2022.

[5] T. Sodikin, Luthfia; Putri, Tineke Fatma; Hidayat, "Analisa Kompresi File Teks Menggunakan Algortima Huffman," J. ICTEE, vol. 3, no. 1, pp. 10–19, 2022.

[6] A. Pahdi, "Algoritma Huffman Dalam Pemampatan Dan Enksripsi Data," Indones. J. Netw. Secur., vol. 6, no. 3, pp. 1–7, 2017.

[7] S. dkk Siboro, "Application of Huffman algorithm and Unary Codes for Text File Compression," Sink. J. dan Penelit. Tek. Inform., vol. 7, no. 3, pp. 1000–1007, 2022.

[8] R. Arshad, A. Saleem, dan D. Khan, "Performance comparison of Huffman Coding and Double Huffman Coding," in *2016 6th International Conference on Innovative Computing Technology, INTECH 2016*, 2017, hal. 361–364. doi: 10.1109/INTECH.2016.7845058.

[9] M. U. Hassan, M. H. Rehmani, dan J. Chen, *Huff-DP: Huffman Coding based Differential Privacy Mechanism for Real-Time Data*. 2023. doi: 10.48550/arXiv.2301.10395.

e-ISSN 2827-8356
p-ISSN 2827-9999

[10] S. More, V. Sanivarapu, Y. Sharma, V. M. Thigale, dan M. Suguna, "Enhancing Data Compression: A Dynamic Programming Approach with Huffman Coding and Burrows-Wheeler Transform," in *2nd International Conference on Automation, Computing and Renewable Systems, ICACRS 2023 - Proceedings*, 2023, hal. 82–88. doi: 10.1109/ICACRS58579.2023.10404627.