Jurnal Pendidikan Teknologi Informasi Dan Komputer
Volume 02, No 01, Mei 2023 Page: 33−38
ISSN 2962-3936 (media online)
https://journal.grahamitra.id/index.php/petik

Penggunaan Algoritma Huffman Dalam Mengkompres Data (File Text)

Raudatul Jannah[1*], Ilham Yani Putra[2], Hayatul Fikri[3], Syaiful Bilad[4]

[1,2,3,4]Fakultas Keguruan dan Ilmu Pendidikan, Program Studi Matematika, UIN Sjech M. Djamil Djambek, Bukittinggi, Indonesia
Email: [1]jannahraudhatull21@gmail.com, [2]yayanilham160@gmail.com , [3]syaifulbilad10@gmail.com
Email Penulis Korespondensi: emailpenuliskorespondensi@email.com
Submitted: **16/05/2023**; Accepted: **23/05/2023**; Published**: 30/05/2023**

Abstrak− Kompresi data merupakan sebuah teknik untuk memperkecil sebuah ukuran data dengan data aslinya. Pemampatan data biasanya diterapkan pada mesin komputer, hal ini dikarenakan setiap simbol yang terdapat pada komputer memiliki nilai nilai bit yang berbeda. Pemampatan data dapat digunakan untuk mengurangkan jumlah bit yang dihasilkan dari setiap simbol yang muncul, dengan itu diharapkan untuk dapat mengurangi ukuran dalam ruang penyimpanan. Pemampatan data dilakukan dengan mengkodekan setiap karakter di dalam arsip dikodekan dengan yang lebih pendek.

Kata Kunci: Algoritma; Kompresi; File text

Abstract−Data compression is a technique to reduce a data size with the original data. Data compression is usually applied to computer machines, this is because each symbol contained on a computer has a different bit value. Data compression can be used to reduce the number of bits generated from each symbol that appears, with the hope of reducing the size of the storage space. Data compression is done by encoding each character in the encoded file with a shorter one.

Keywords: Algorithm; Compression; Text files

1. PENDAHULUAN

Dalam penyimpanan data berbentuk arsip (file text) yang berukuran besar akan memakan ruang penyimpanan yang besar juga. Hal ini dapat diatasi dengan mengkodekan isi arsip sesingkat mungkin, sehingga ruang penyimpanan yang dibutuhkan juga sedikit. Cara pengkodean seperti ini disebut pemampatan (compression) data.

Pemampatan data dilakukan dengan mengkodekan setiap karakter di dalam arsip dikodekan dengan kode yang lebih pendek. Sistem kode yang banyak digunakan adalah kode ASCII (American Standard Code for Information Interchange). Dengan kode ASCII, setiap karakter dikodekan dalam 8 bit biner [10]. Untuk meminimumkan jumlah bit yang dibutuhkan, panjang kode untuk setiap karakter sedapat mungkin diperpendek, terutama untuk karakter yang kekerapan (frequency) kemunculannya besar. Pemikiran seperti inilah yang mendasari munculnya kode Huffman.

Algoritma Huffman, dibuat pertama kali oleh Prof. David A. Huffman (1925-1999) pada tahun 1952 sebagai desertasi Ph.D dengan publikasi berjudul ”A Method for the Construction of Minimum-Redundancy Codes”. Algoritma Huffman menggunakan prinsip bahwa setiap karakter dikodekan dengan serangkaian bit biner, dimana karakter yang sering muncul dikodekan dengan rangkaian bit yang pendek dan karakter yang jarang muncul dikodekan dengan rangkian bit yang lebih panjang.

Tujuan dari penulisan ini adalah untuk mengetahui keefektifan algoritma Huffman dalam mengkompres data (file text). Pada penulisan ini kami membahas langkah-langkah penggunaan algoritma Huffman dan efisiensinya

2. METODOLOGI PENELITIAN

Penelitian ini difokuskan pada pemampatan data yang dilakukan dengan mengkode setiap karakter untuk dapat mengurangi ukuran dalam sebuah data. Hal ini dapat diatasi dengan mengkode isi arsip dengan sesingkat mungkin diperpendek. Sistem kode yang digunakan adalah kode ASCII (American Standard Code For Information Interchange). Dengan menggunakan kode ASCII setiap karakter dapat diberi kode dalam 8 bit biner. Algoritma huffman menggunakan prinsip bahwa setiap karakter dikodekan dengan serangkaian 8 bit biner, dimana karakter yang sering muncul dikodekan dengan 8 bit yang pendek sehingga karakter yang jarang muncul dikodekan dengan rangkaian bit yang lebih panjang.

3. HASIL DAN PEMBAHASAN

3.1 Algoritma Huffman

Irwan Wardoyo, dkk menjelaskan dalam papernya bahwa Algoritma Huffman merupakan salah satu algoritma yang digunakan untuk mengompres teks. Berdasarkan teknik pengkodeannya, algoritma Huffman menggunakan metode symbolwise, yaitu menghitung peluang kemunculan dari setiap karakter, dimana karakter yang sering muncul diberi

Jurnal Pendidikan Teknologi Informasi Dan Komputer
Volume 02, No 01, Mei 2023 Page: 33−38
ISSN 2962-3936 (media online)
https://journal.grahamitra.id/index.php/petik

kode yang lebih pendek dibandingkan dengan karakter yang jarang muncul. Dalam prosesnya terdapat tiga tahap yaitu :

a. Tahap pembentukan pohon Huffman
   Kode Huffman merupakan kode prefiks yang berisi kumpulan kode biner, dimana pada kode prefiks ini tidak ada kode biner yang menjadi awal bagi kode biner yang lain. Kode ini biasanya identik dengan pohon biner yang diberi label 0 untuk cabang kiri dan 1 untuk cabang kanan. Langkah – langkah pembentukan pohon Huffman adalah sebagai berikut:
   1. Membaca semua karakter yang ada di teks. Setiap karakter dinyatakan sebagai pohon bertitik tunggal.
   2. Menggabungkan dua pohon yang mempunyai frekuensi terkecil pada sebuah akar. Setelah digabungkan akar tesebut akan mempunyai frekuensi yang merupakan jumlah dari dua pohon penyusunnya. Mengulangi proses tersebut sampai tersisa satu pohon Huffman.
b. Proses Encoding
   Encoding ialah cara menyusun string biner dari teks. Prosesnya dimulai dengan membuat pohon Huffman. Setelah itu, kode untuk satu karakter dibuat dengan menyusun nama string biner yang dibaca dari akar sampai daun pohon Huffman. Langkahnya yaitu menentukan karakter yang akan di-encoding. Kemudian baca setiap bit yang ada pada cabang sampai ketemu daun dimana karakter itu berada sampai seluruh karakter di-encoding.
c. Proses Decoding
   Decoding ialah menyusun kembali data dari string biner menjadi karakter kembali. Langkahnya yaitu membaca sebuah bit dari string biner dimulai dari akar. Kemudian kodekan rangkaian bit yang telah dibaca menjadi karakter sampai semua bit di dalam string habis.

3.2 Mengkompresi Data Menggunakan Algoritma Huffman

Pemampatan data dilakukan dengan menkodekan setiap karakter di dalam pesan atau di dalam arsip dikodekan dengan kode yang lebih pendek. Dengan kode ASCII, setiap karakter dikodekan dalam 8 bit biner. Seperti terlihat pada tabel berikut.
Contoh 1

Tabel 1. Kode ASCII

Karakter	Kode ASCII
A	01000001
B	01000010
C	01000011
D	01000100

Dengan mengikuti ketentuan pengkodean di atas, string ‘ABACCDA’ dipresentasikan menjadi rangkaian bit:
01000001010000100100000101000011010000110100010001000001
Jadi, dengan sistem pengkodean ASCII, representasi 7 huruf membutuhkan 7 × 8 = 56 bit. Untuk meminimumkan jumlah bit yang dibutuhkan, panjang kode untuk setiap karakter sedapat mungkin diperpendek, terutama untuk karakter yang kekerapan (frequency) kemunculannya besar. Pada pesan ‘ABACCDA’, kekerapan kemunculan setiap karakternya dapat dilihat pada tabel berikut.

Tabel 2. Kekerapan Kemunculan Karakter

No.	Karakter	Kekerapan
1	A	3
2	B	1
3	C	2
4	D	1

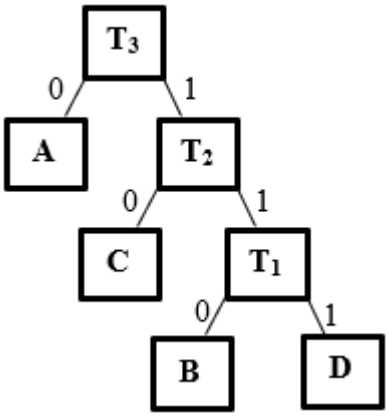
Untuk mendapatkan kode Huffman, kita harus menghitung dulu frkuensi kemunculan setiap karakter di dalam teks dengan mengikuti langkah-langkah pada algoritma Huffman sebagai berikut.
LANGKAH 1. Pembentukan Pohon Huffman

Tabel 3. Tabel Pembentukan Pohon Huffman

Index	1	2	3	4	5	6	7
Titik	A	B	C	D	T_1	T_2	T_3
Frek	3	1	2	1	2	4	7
Tree	T_3	T_1	T_2	T_1	T_2	T_3	-
Kode	0	0	0	1	1	1	-

Bedasarkan tabel di atas, akan dihasilkan pohon Huffman sebagai berikut.

Jurnal Pendidikan Teknologi Informasi Dan Komputer
Volume 02, No 01, Mei 2023 Page: 33−38
ISSN 2962-3936 (media online)
https://journal.grahamitra.id/index.php/petik



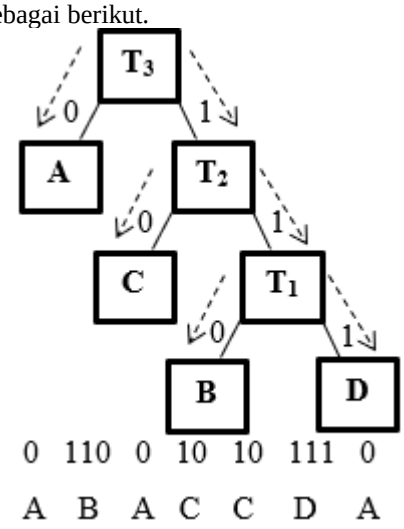
Gambar 1. Pohon Huffman

LANGKAH 2. Proses Encoding
Proses encoding dimulai dengan membuat pohon Huffman terlebih dahulu. Setelah itu kode untuk satu karakter dibuat dengan menyusun nama string biner yang dibaca dari akar sampai ke daun pada pohon Huffman. Dapat dilihat hasil encoding dari pohon Huffman pada tabel berikut.

Tabel 4. Hasil konversi karakter ke dalam kode Huffman

No.	Karakter	Kekerapan	Kode Huffman
1	A	3	0
2	B	1	110
3	C	2	10
4	D	1	111

LANGKAH 3. Proses Decoding
Decoding adalah kebalikan dari proses Encoding, yang berarti penyusunan kembali data dari string biner menjadi data semula. Dengan menggunakan pohon Huffman pada gambar 3.1 akan didecoding setiap bit pada string ‘ABACCDA’. Proses Decoding dapat digambarkan sebagai berikut.



Gambar 2. Proses Decoding kode Huffman

Berdasarkan langkah-langkah di atas dengan menggunakan kode Huffman, pesan ‘ABACCDA’ dipresentasikan menjadi rangkaian berikut.

0110010101110

Jadi, dengan menggunakan kode Huffman, jumlah bit yang dibutuhkan untuk string ‘ABACCDA’ hanya 13 bit. Karakter yang sering muncul di dalam string dipresentasikan dengan kode yang lebih pendek daripada kode untuk karakter yang jarang muncul.

Contoh 2
Sebuah teks yang berisi 100.000 string, diantaranya 45.000 karakter ‘g’, 13. 000 karakter ‘o’, 12.000 karakter ‘p’, 16. 000 karakter ‘h’, 9.000 karakter ‘e’, dan 5.000 karakter ‘r’. berdasarkan string tersebut diketahui tabel kode ASCII sebagai berikut:

Jurnal Pendidikan Teknologi Informasi Dan Komputer
Volume 02, No 01, Mei 2023 Page: 33−38
ISSN 2962-3936 (media online)
https://journal.grahamitra.id/index.php/petik

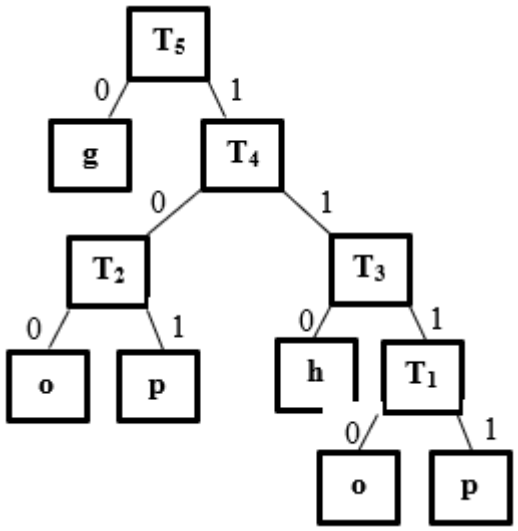
Tabel 5. Kode ASCII dan Frekuensi

Karakter	Kode ASCII	Frekuensi
g	01100111	45.000
o	01101111	13.000
p	01110000	12.000
h	01101000	16.000
e	01100101	9.000
r	01110010	5.000

Dengan menggunakan kode ASCII, kita membutuhkan memori sebesar:
45000 × 8 bit (01100111) = 360.000 bit
13.000 × 8 bit (01101111) = 104.000 bit
12.000 × 8 bit (01110000) = 96.000 bit
16.000 × 8 bit (01101000) = 128.000 bit
9.000 × 8 bit (01100101) = 72.000 bit
5000 × 8 bit (01110010) = 40.000 bit
Jumlah = 800.000 bit
Kita akan membandingkan besar memori yang dibutuhkan jika menggunakan algoritma Huffman.
LANGKAH 1. Pembentukan Pohon Huffman

Tabel 6. Tabel Pembentukan Pohon Huffman

Index	1	2	3	4	5	6	7	8	9	10	11
Titik	g	o	p	h	e	r	T_1	T_2	T_3	T_5	T_6
Frek (Ribuan)	45	13	12	16	9	5	14	25	30	55	100
Tree	T_5	T_2	T_2	T_3	T_1	T_1	T_3	T_4	T_4	T_5	-
Kode	0	0	1	0	0	1	1	0	1	1	-



Gambar 3. Pohon Huffman

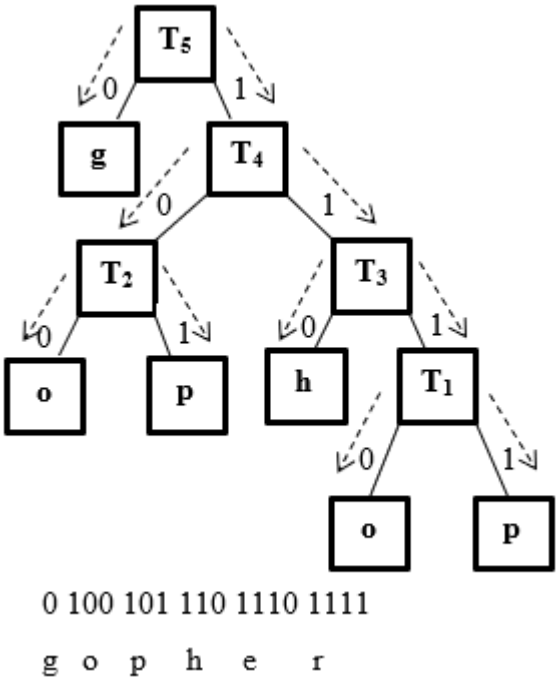
LANGKAH 2. Proses Encoding

Tabel 7. Hasil konversi karakter ke dalam kode Huffman

No.	**Karakter**	**Frekuensi**	**Kode Huffman**
1	g	45.000	0
2	o	13.000	100
3	p	12.000	101
4	h	16.000	110
5	e	9.000	1110
6	r	5.000	1111

LANGKAH 3. Proses Decoding

Jurnal Pendidikan Teknologi Informasi Dan Komputer
Volume 02, No 01, Mei 2023 Page: 33−38
ISSN 2962-3936 (media online)
https://journal.grahamitra.id/index.php/petik



Gambar 4. Proses Decoding kode Huffman

Berdasarkan langkah-langkah di atas dengan menggunakan kode Huffman, karakter g, o, p, h, e dan r dipresentasikan menjadi rangkaian berikut.
45.000 × 1 bit (0) = 45.000 bit
13.000 × 3 bit (100) = 39.000 bit
12.000 × 3 bit (101) = 36.000 bit
16.000 × 3 bit (110) = 48.000 bit
9. 000 × 4 bit (1110) = 36.000 bit
5000 × 4 bit (1111) = 20.000 bit
Jumlah = 224. 000 bit
Berdasarkan jumlah memori yang dibutuhkan menggunakan algoritma Huffman, menunjukkan bahwa dengan menggunakan algoritma Huffman kita dapat menghemat besar memori pada suatu data, sehingga efisien untuk digunakan.

4. KESIMPULAN

Algoritma Huffman merupakan salah satu algoritma yang digunakan untuk mengompres teks. Berdasarkan teknik pengkodeannya, algoritma Huffman menggunakan metode symbolwise, yaitu menghitung peluang kemunculan dari setiap karakter, dimana karakter yang sering muncul diberi kode yang lebih pendek dibandingkan dengan karakter yang jarang muncul. Dalam prosesnya terdapat tiga tahap yaitu: 1) Tahap pembentukan pohon Huffman, Kode Huffman merupakan kode prefiks yang berisi kumpulan kode biner, dimana pada kode prefiks ini tidak ada kode biner yang menjadi awal bagi kode biner yang lain. Kode ini biasanya identik dengan pohon biner yang diberi label 0 untuk cabang kiri dan 1 untuk cabang kanan. Langkah-langkah pembentukan pohon, yaitu: a) Membaca semua karakter yang ada di teks. Setiap karakter dinyatakan sebagai pohon bertitik tunggal, b) Menggabungkan dua pohon yang mempunyai frekuensi terkecil pada sebuah akar. Setelah digabungkan akar tesebut akan mempunyai frekuensi yang merupakan jumlah dari dua pohon penyusunnya. Mengulangi proses tersebut sampai tersisa satu pohon Huffman. 2) Proses Encoding, Encoding ialah cara menyusun string biner dari teks. Prosesnya dimulai dengan membuat pohon Huffman. Langkahnya yaitu menentukan karakter yang akan di-encoding. Kemudian baca setiap bit yang ada pada cabang sampai ketemu daun dimana karakter itu berada sampai seluruh karakter di-encoding. 3) Proses Decoding, Decoding ialah menyusun kembali data dari string biner menjadi karakter kembali. Langkahnya yaitu membaca sebuah bit dari string biner dimulai dari akar. Kemudian kodekan rangkaian bit yang telah dibaca menjadi karakter sampai semua bit di dalam string habis.

REFERENCES

[1] Anonim. (2008). Bilangan Biner (Online).
[2] Bell, Timothy C, Dkk. (1990). Text Compression, Prentice Hall, Englewood NJ.

[3] Blelloch, G.E. (2001). Introduction to Data Compression. Computer Science Departemen, Carnegie Mellon University.
[4] Cormen, T.H, Charles E.L, and Ronald L.R . (2001). Introduction to Algoritms. Second Edition, London; Mc Graw-Hill Book Company.
[5] Halvsorson Michael. (2000). Microsoft Visual Basic 6.0 Step By Step. Jakarta; PT. Elex Media Komputindo.
[6] Haryanto, R.I. (2009). Kompresi Data dengan Algoritma Huffman dan Perbandingannya dengan Algoritma LZW dan DMC. Strategi Algoritmik.
[7] Irwan Wardoyo, Peri Kusdinar & Irvan Hasbi Taufik. (n.d.). Kompresi Teks dengan Menggunakan Algoritma Huffman.
[8] Irwan Wardoyo, Peri Kusdinar, Irvan Hasbi Taufik. (2012). Kompresi Teks dengan Menggunakan Algoritma Huffman. Jurnal Ilmiah, Vol. 5 No.2(Fakultas Teknologi Informasi Institut Teknologi Sepuluh November (its) Surabaya), 3-6.
[9] Munir, Rinaldi. (n.d.). Matematika Diskrit. Informatika.
[10] Rinaldi Munir. (2012). Matematika Diskrit. Bandung. Informatika.
[11] Samuel Wibisno. (2008). Matematika Diskrit. Yogyakarta. Graha Ilmu.
[12] Silalahi, B.P, Julio A and Danny D.S. (n.d.). Perbandingan ALgoritma Huffman Statik dengan Algoritma Adaptif pada Kompresi Teks. Jurnal Penelitian.
[13] Susda Heleni & Zulkarnain. (2006). Matematika Diskrit. Pekanbaru. Universitas Riau.
[14] Torer, J. A. (1998). Data Compression, Computer Science Press, Rockville, MD Witten, Ian H., Neal, Radford M.