

# Aplikasi Huffman Coding dalam Kompresi File

Leonardo Gianto\*, Michael Fransiscus Munthe<sup>†</sup>, Nikolas<sup>‡</sup>, Aditya Anandita Dharma Putra<sup>§</sup>

*School of Electrical Engineering and Informatics*

*Institut Teknologi Bandung*

Bandung, Indonesia

{\*13219001, <sup>†</sup>13219029, <sup>‡</sup>13219041, <sup>§</sup>13219043}@std.stei.itb.ac.id

**Abstract**—Paper ini mengreview dan implementasi aplikasi Huffman encoding dalam kompresi file text. Paper dilengkapi dengan kode C implementasi program yang dibahas dan contoh penggunaannya dalam mengkompresi sebuah file text. Hasil yang didapatkan adalah metode Huffman encoding berhasil mengompresi file text menjadi 57 persen ukuran awalnya. Hasil encoding kemudian juga harus dapat kembali didecode ke text awalnya. Hasil yang didapatkan adalah decoding memiliki akurasi 100 persen atau berhasil menghasilkan file decoded yang utuh.

**Index Terms**—Huffman, encoding, decoding, review

## I. INTRODUCTION

Pada paper ini terdapat pembahasan dan review mengenai paper yang disebutkan pada referensi [1]. Adapun source code pada dalam implementasi paper ini berasal dari referensi [2]. Konten dari review pada paper ini meliputi ringkasan singkat isi, problem, pendekatan, kekurangan dan saran-saran untuk memperbaiki. Paper juga dilengkapi dengan implementasi kode C program.

## II. RINGKASAN SINGKAT

Kompresi adalah metode mengurangi jumlah data yang digunakan untuk menyimpan sebuah file tanpa mengurangi kualitas dan keutuhan file tersebut secara berlebihan. Terdapat dua buah teknik kompresi data termasuk Arithmetic Coding dan Huffman Coding. Kompresi ini termasuk pada teknik kompresi lossless (tidak ada data terbuang) dan menghasilkan output sebuah kode dari sebuah string karakter.

Cara kerja kompresi Huffman coding adalah menggunakan sebuah frekuensi kemunculan sebuah data item. Dari data frekuensi tersebut disusun sebuah variable-length code table. Karakter dengan kemunculan terbanyak dinotasikan dengan kode terpendek dan karakter dengan kemunculan paling sedikit dinotasikan dengan kode terpanjang. Dengan demikian Huffman coding menjadi metode kompresi data paling efisien.

## III. PROBLEM

Problem yang dihadapi saat ini adalah di masa digital, kecepatan dan efisiensi menjadi hal yang tidak dapat lagi diabaikan. Data-data yang besar dan berat mulai ditinggalkan dan digantikan dengan data yang ringan namun tidak mengurangi kualitas informasi yang terkandung.

Berbagai metode dari waktu ke waktu berdatangan memberikan cara untuk mengkompresi data tersebut menjadi semakin ringan tanpa menghilangkan keutuhan. Teknik-teknik kompresi yang kalah efisien akan selalu tergantikan.

Huffman Coding sebagai salah satu teknik kompresi data yang paling efisien telah ada sejak tahun 1951, dan metode ini sampai sekarang masih relevan karena sederhana dan dapat diaplikasikan ke banyak hal termasuk text, mp3, dan citra (gambar).

## IV. PENDEKATAN

Pendekatan penyelesaian menggunakan metode Huffman dapat dijelaskan sebagai berikut (secara basic):

Asumsikan sebuah sumber menghasilkan 4 buah simbol (a, b, c, d) dengan probabilitas kemunculannya secara berurutan: (0.4, 0.35, 0.2, 0.05), dibuat sebuah binary tree dari kiri ke kanan (yang paling jarang ke yang paling mungkin) dengan mengambil 2 simbol paling jarang dan membuat sebuah simbol baru yang probabilitasnya penjumlahan kedua sinyal tersebut. Hal ini terus dilakukan sampai hanya terdapat 1 buah simbol. Tree dibaca berkebalikan dari kanan ke kiri, menghasilkan:

TABLE I  
HUFFMAN CODE CONTOH ABCD

Character	Code
a	0
b	10
c	111
d	011

Teknik ini dilakukan dengan membuat sebuah binary tree dengan nodes yang dapat disimpan dalam array biasa dengan besar sesuai jumlah simbol (N). Semua node awalnya adalah leaf node dan bisa menjadi internal node. Sebuah tree yang komplet memiliki N buah leaf node dan N-1 buah internal node.

Langkah-langkah dalam pembuatan tree:

- 1) Mulai dengan membuat leaf sejumlah simbol yang ada.
- 2) Urutkan seluruh leaf node ke queue pertama, simbol dengan probabilitas terkecil di awal.
- 3) Apabila ada lebih dari 1 node di queue:
  - a) Dequeue dua node terkecil.
  - b) Buat sebuah internal node dan kedua node terkecil tadi.
  - c) Masukkan node tadi ke ujung queue kedua.
- 4) Node yang tersisa adalah root node, tree selesai.

Metode ini memiliki performance linear-time apabila leaf node sudah tersortir, jika tidak maka time complexitynya adalah  $O(n \log n)$ .

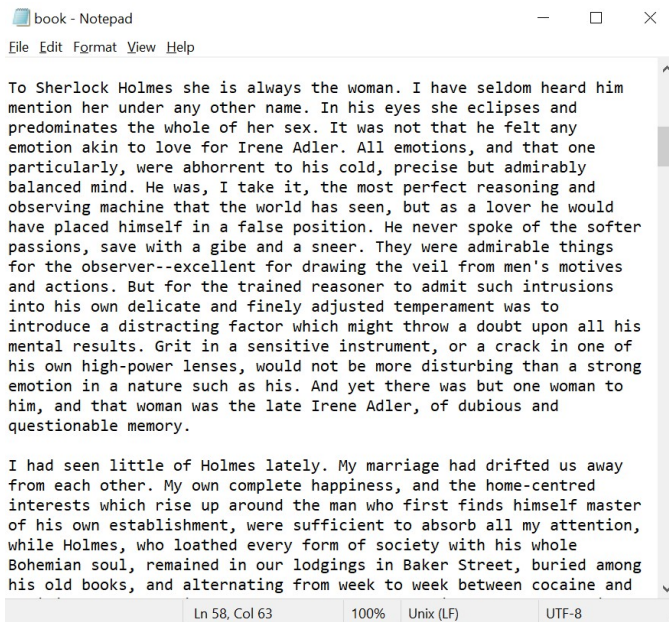


Fig. 1. File text cuplikan buku yang akan dikompresi.

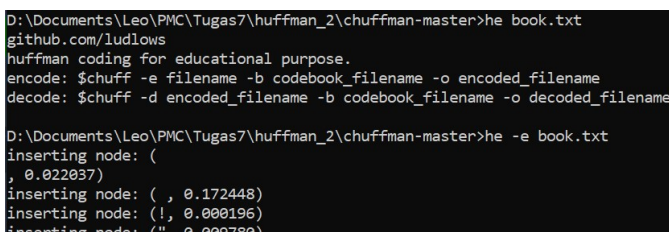


Fig. 2. Eksekusi program menggunakan command prompt.

## V. IMPLEMENTASI MENGGUNAKAN C

Implementasi program menggunakan bahasa pemrograman C dengan source code program encode dan decode terlampir bersama file ini.

### A. Kompresi External Text File

1) *Proses Encoding*: Berikut adalah contoh aplikasinya dalam mengompresi sebuah file text berisi text cuplikan dari buku Sherlock Holmes:

File text book.txt pada **Fig. 1** dimasukkan ke program. Setelah dilakukan input, program langsung membentuk node-node Huffman nya seperti dapat dilihat pada **Fig. 2**. Dari frekuensi kemunculan karakter-karakter pada text terbentuk sebuah Huffman table pada **Fig. 3**, sehingga kompresi dengan metode Huffman menghasilkan kompresi yang paling efisien. Program ini menghasilkan juga output Huffman table yang dipakai dalam sebuah file text external codebook.txt. Hasil kompresi dapat dilihat pada **Fig. 4** berupa binary code.

Apabila space complexity kedua buah file dihitung, didapatkan bahwa file awal membutuhkan space 122704 bit, sementara hasil Huffman encodingnya membutuhkan 69563 bit. Dalam kata lain, kompresi Huffman berhasil mengecilkan file dengan compression rate sampai dengan 56.69 persen.

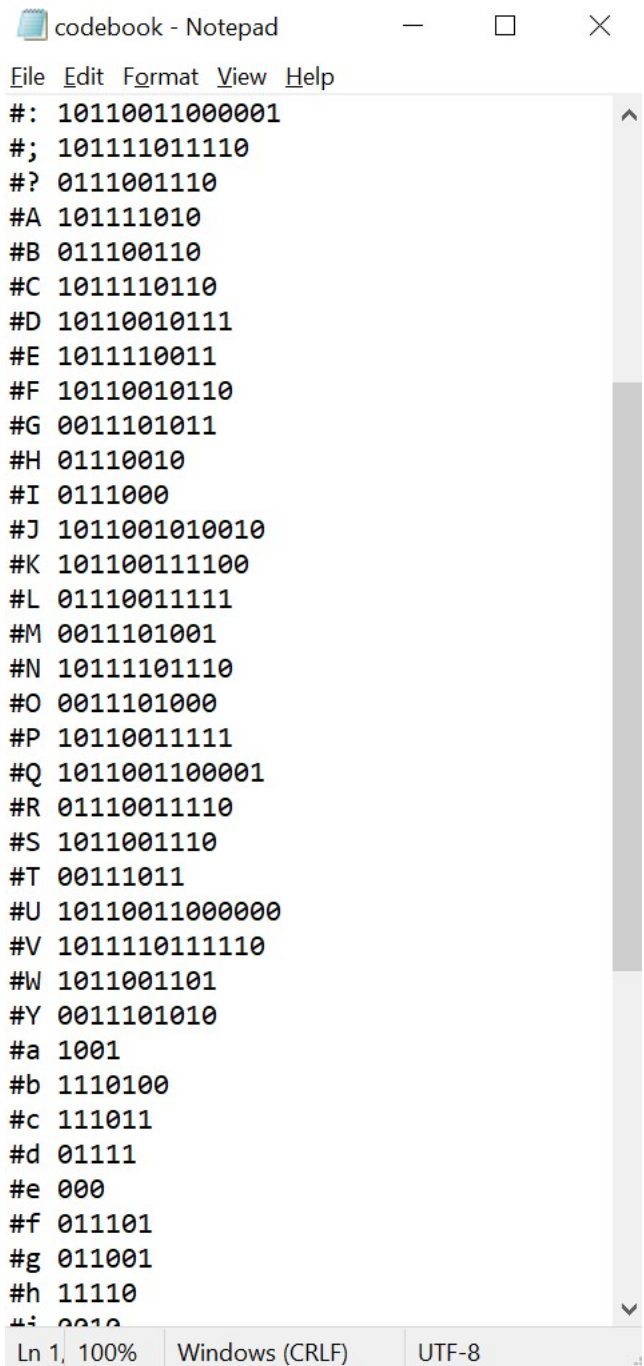


Fig. 3. Huffman table yang terbentuk dari file book.txt awal tersimpan dalam file codebook.txt.

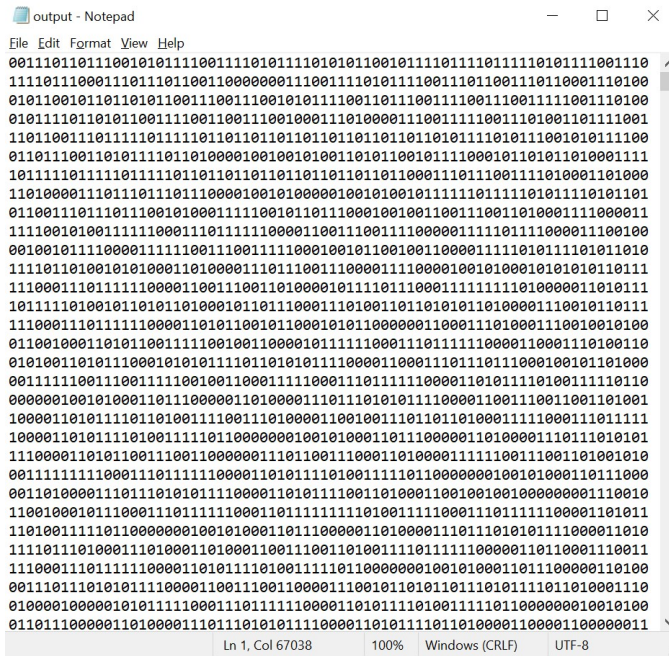


Fig. 4. Hasil encoding file text menggunakan Huffman.

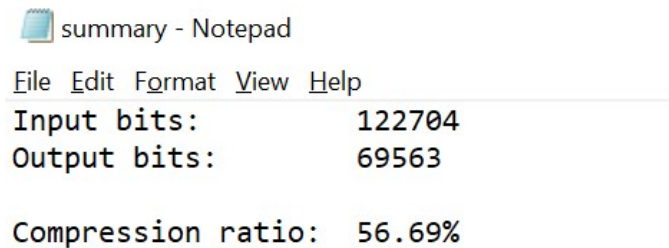


Fig. 5. Encoding menggunakan Huffman mengompresi file sampai 56.69 persen.

2) *Proses Decoding*: Berikut adalah decoding dari kode pada **Fig. 4**: Pertama-tama dilakukan eksekusi program dan input file yang telah terencode dalam sebuah file dinamakan *encoded.txt* ke program seperti pada **Fig. 6**.

Hasil decoding tersimpan dalam sebuah file text eksternal juga yang dinamakan *source.txt* seperti dapat dilihat pada **Fig. 7**.

Output hasil decoding kembali yang tersimpan pada file *origin.txt* utuh dan tidak terdapat cacat (error).

Dapat disimpulkan bahwa untuk file *book.txt*, program

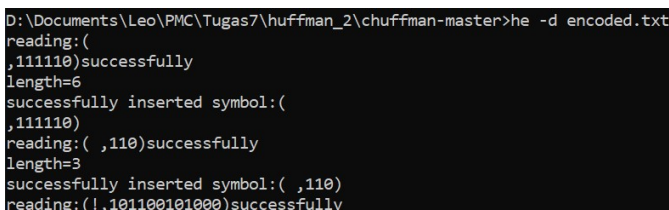


Fig. 6. Input file encoded .huffman yang akan di decode.

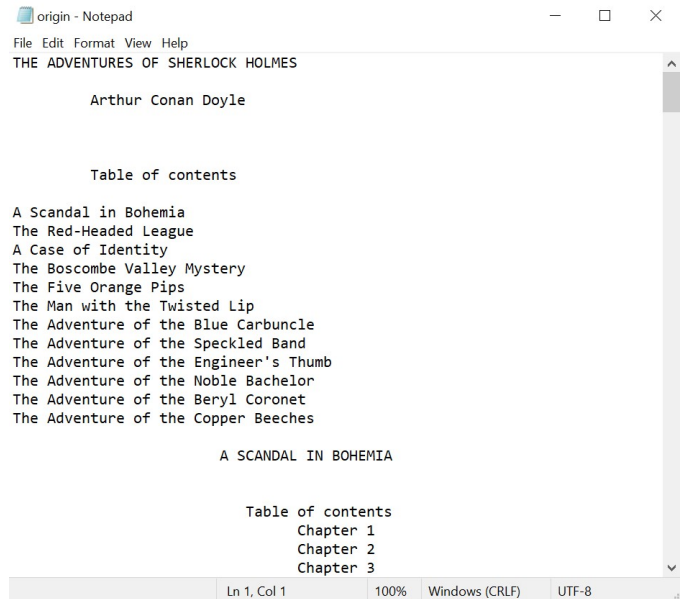


Fig. 7. Hasil decoding Huffman-encoded input.

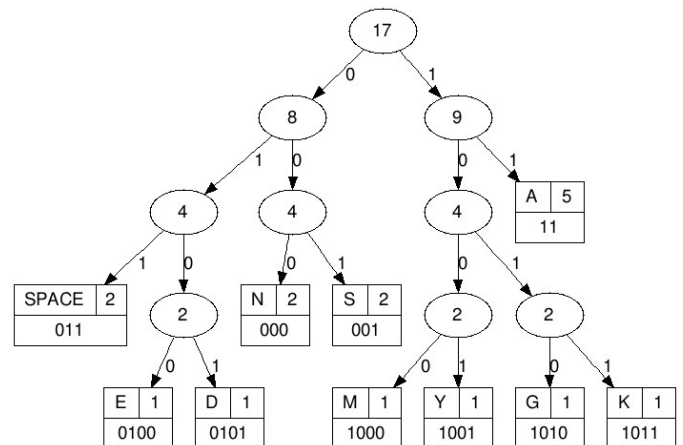


Fig. 8. Huffman Tree dari kata "saya sedang makan"

Huffman Compression memiliki efektivitas 56.69 persen dan akurasi 100 persen.

## B. Perbandingan dengan Perhitungan Tangan

1) *Perhitungan Tangan*: Pada bagian ini akan dilakukan perbandingan hasil kompresi menggunakan tangan dengan menggunakan kode.

Pertama-tama asumsikan sebuah kalimat yang akan dikompresi:

*saya sedang makan*

Tabel frekuensi kemunculan karakter dan Huffman table dari string *saya sedang makan* ditunjukkan pada **Fig. 9**.

Maka apabila digabungkan, hasil encoding text "saya sedang makan" adalah:



No.	Huruf	Code	Average bit Length
1.	E	0100	0.0588235
2.	D	0101	0.0588235
3.	M	1000	0.0588235
4.	Y	1001	0.0588235
5.	G	1010	0.0588235
6.	K	1011	0.0588235
7.	N	000	0.117647
8.	S	001	0.117647
9.	Space	011	0.117647
10.	A	11	0.294118
Total			1

Fig. 9. Tabel Huffman dari "saya sedang makan"

```
D:\Documents\Leo\PKC\Tugas7\huffman_2\chuffman-master>he -e makan.txt -b codemakan.txt -o encodedmakan.txt
inserting node: ( , 0.117647)
inserting node: (a, 0.294118)
inserting node: (d, 0.058824)
inserting node: (e, 0.058824)
inserting node: (g, 0.058824)
inserting node: (k, 0.058824)
inserting node: (m, 0.058824)
inserting node: (n, 0.117647)
inserting node: (s, 0.117647)
inserting node: (y, 0.058824)
```

Fig. 10. Hasil eksekusi program dengan input text saya sedang makan dalam file eksternal.

```
001 11 1001 11 011 001 0100 0101 11 000
1010 011 1000 11 1011 11 000
```

Text saya sedang makan memiliki ukuran 136 byte, sementara hasil encoding Huffmannya memiliki ukuran 52 bit.

2) *Perhitungan Program*: Pada bagian ini akan dilakukan kompresi text saya sedang makan menggunakan program. Berikut adalah command run program dan inputnya pada **Fig. 10**.

Program langsung membentuk node-node dan mengonstruksi sebuah Huffman Table dalam sebuah file eksternal *codemakan.txt* seperti pada **Fig. 11**.

Didapatkan hasil bahwa input memiliki ukuran 136 bit dan hasil encoding memiliki ukuran 52 bit (compression rate 38.24 persen) dapat dilihat pada **Fig. 12**.

Berikutnya dilakukan decoding kembali ke text awal, pertama-tama dilakukan eksekusi program dan input text encoded. **Fig. 13**.


Didapatkan seperti pada **Fig. 14** bahwa hasil decoding file encoded ditempatkan dalam sebuah file *originmakan.txt* berhasil menghasilkan sebuah file yang kontennya utuh sesuai dengan inputnya (tanpa error).

## VI. KESIMPULAN

Dari hasil analisis di atas diperoleh kesimpulan sebagai berikut.

Yang pertama, teknik kompresi Huffman memiliki compression rate rata-rata 57 sampai 38 persen ukuran file awal.

Yang kedua, kompresi Huffman baik untuk file kecil maupun file besar memiliki akurasi 100 persen, sementara efektivitas kompresi file kecil lebih tinggi daripada file besar. Hal ini dikarenakan juga bahwa pada file besar terdapat banyak karakter-karakter yang unik termasuk huruf kapital, tanda baca, space, dan baris baru. Sementara itu dalam file kecil hanya terdapat 3 buah kata dan tidak ada karakter-karakter unik.

 codemakan - Notepad

File Edit Format View Help

```
# 001
#a 11
#d 1001
#e 1000
#g 1011
#k 1010
#m 0101
#n 000
#s 011
#y 0100
```

Fig. 11. Huffman Table yang dibentuk program.

 summarymakan - Notepad

File Edit Format View Help


```
Input bits: 136
Output bits: 52
```

Compression ratio: 38.24%

Fig. 12. Encoding text saya sedang makan menggunakan program menghasilkan compression rate 38.24 persen.

```
D:\Documents\Leo\PKC\Tugas7\huffman_2\chuffman-master>he -d encodedmakan.txt -b codemakan.txt -o originmakan.txt
reading: ( ,001)successfully
length=3
successfully inserted symbol:( ,001)
reading: (a,11)successfully
length=2
successfully inserted symbol:(a,11)
reading: (d,1001)successfully
length=4
successfully inserted symbol:(d,1001)
reading: (e,1000)successfully
```

Fig. 13. Eksekusi program decode dan input teks terencodena.

 originmakan - Notepad

File Edit Format View Help

saya sedang makan

Fig. 14. Hasil output program decode.

## VII. KEKURANGAN

- Efisiensi algoritma berbeda-beda untuk jenis file yang berbeda, namun tidak dapat lebih tinggi dari 8:1 compression.
- Kompresi file citra yang memiliki banyak identical pixel tidak seefisien menggunakan teknik RLE.
- Diharuskan menyertakan sebuah Huffman table bersama compressed file untuk decoding.
- Teknik Huffman mengencode data tidak dalam fixed-length, dan sulit untuk mengenali data EOF. Hal ini mengakibatkan apabila ada sedikit saja data encoded yang corrupt (bit tambahan atau bit hilang) maka seluruh data akan rusak juga.
- Encoding menggunakan teknik Huffman juga dapat terbilang lambat.

## VIII. SARAN

Huffman coding paling banyak dan paling efektif digunakan dalam kompresi file JPEG, namun saat ini sudah banyak sekali perkembangan di bidang kompresi file menggunakan teknik ini. Salah satu contohnya adalah Adaptive Huffman Coding, di mana kode dibuat seiring simbol-simbolnya sedang ditransmisikan. Dengan demikian sumbernya dapat diencode real-time, walaupun semakin sensitif terhadap error di transmisi. Hal ini dapat diaplikasikan untuk mempercepat transmisi data secara langsung.

Selain itu, ke depannya mungkin teknik Huffman encoding dapat diterapkan untuk file MP4 juga. Untuk itu perlu dilakukan perbandingan performa teknik-teknik encoding Huffman, Algorithm, LZW, dan lain sebagainya untuk memperoleh teknik yang paling efektif mengingat banyaknya repetitive pixel/frame dalam file MP4.

## ACKNOWLEDGMENT

Penulis ingin mengucapkan banyak terima kasih kepada Bapak Muhammad Ogin Hasanuddin S.T.,M.T. selaku dosen pada mata kuliah Pemecahan Masalah dengan C yang telah mengajarkan dan membimbing selama pengerjaan tugas paper sehingga tugas ini dapat terselesaikan dengan baik dan memuaskan.

## REFERENCES

- [1] M. Sharma *et al.*, "Compression using huffman coding," *IJCSNS International Journal of Computer Science and Network Security*, vol. 10, no. 5, pp. 133–141, 2010.
- [2] Ludlows, "Codes for Huffman Encode and Decode," 2019. [Online]. Available: <https://github.com/ludlows/chuffman>