Penerapan Kode Huffman untuk Kompresi Data Teks dan Gambar

Muhamad Arif Adiputra (13516114)
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
hotaru@itb.ac.id

***Abstraksi* — Makalah ini berisi tentang penjelasan awal mengenai kompresi data, jenis-jenis kompresi data yang ada, penjelasan mengenai algoritma Huffman, dan penerapan penggunaan kode huffman yang merupakan salah satu algoritma *loseless compression* yang dapat digunakan untuk banyak hal. Dalam makalah ini akan dibahas penggunaan kode huffman pada data teks dan gambar serta bagaimana cara komputer menyimpan data teks tersebut beserta pengaplikasiannya.**

Kata Kunci* — Kompresi data (*data compression),* Kode Huffman, Pemampatan data tanpa kehilangan (*loseless compression), Text Encoding

I. PENDAHULUAN

Dalam pemrosesan data dan sinyal, kompresi data atau reduksi *bit-rate* menyandikan informasi menggunakan bit yang lebih sedikit daripada keadaan awal data tersebut. Kompresi data ini sangat diperlukan di dunia nyata terutama pada era digital ini karena hampir seluruh data yang digunakan dalam kehidupan pada zaman sekarang menggunakan data digital sebagai kebutuhan utama atau kebutuhan tambahan dalam kehidupan sehari-hari.

Pemampatan data atau kompresi data merupakan sebuah cara untuk memadatkan data sehingga menggunakan ruang penyimpanan yang lebih kecil sehingga lebih efisian dalam penyimpanan dan dapat mempersingkat waktu pertukaran data dengan *interface* atau koneksi yang lambat. Terdapat dua jenis pemampatan data :

1. Pemampatan tanpa kehilangan (lossless data compression)
2. Pemampatan berkehilangan (lossy data compression)

Lossless compression merupakan salah satu jenis algoritma kompresi data yang dapat mengompresi data tanpa ada kehilangan atau degradasi dari data original. Kompresi jenis ini digunakan dalam banyak aplikasi. Beberapa contoh diantaranya adalah menggunaan format *file* ZIP dalam kompresi data komputer. Jenis ini digunakan dalam kasus ketika data tersebut penting untuk disimpan dalam keadaan original tanpa ada kehilangan data atau degradasi data dan harus identik dengan data pada keadaan awal sebelum dikompresi dimana deviasi, degradasi, atau kehilangan sedikitpun data dapat menyebabkan data yang akan diambil menjadi tidak berguna. Contoh umum kasus penggunaan *lossless compression* adalah program eksekusi, teks dokumen, kode sumber (*source code)* pemrograman, dan lainnya. Beberapa jenis format gambar seperti PNG dan GIF hanya menggunakan kompresi ini sedingkan yang lainnya seperti TIFF dan MNG dapat menggunakan kedua jenis kompresi.

Sebaliknya, *lossy data compression* merupakan kompresi data dengan deviasi, degradasi, atau kehilangan data yang minim dapat diterima dan tidak mengganggu hasil yang akan diterima. Kasus seperti ini dapat ditemukan di audio, video, dan gambar yang degradasi dengan persentase yang minim dapat diterima.

Pada makalah ini, akan dibahas tentang kode huffman yang merupakan kompresi data tipe *lossless compression*, *text encoding*, dan pengaplikasiannya.

II. KOMPRESI DATA

Terdapat dua jenis kompresi data yang umum digunakan dalam dunia komputer yaitu pemampatan tanpa kehilangan (lossless data compression) dan pemampatan berkehilangan (lossy data compression).

1. Pemampatan tanpa kehilangan (lossless data compression)

Teknik ini dapat memadatkan data dan mengembalikannya sama persis seperti semula. Tidak ada informasi yang hilang, deviasi informasi, ataupun reduksi informasi dari data semula. Biasanya algoritma pemadatan data jenis ini menggunakan prinsip *statistical redudancy* (kelebihan statistik) supaya data dapat disimpa dengan lebih ringkas. Karena banyaknya data yang dipakai sehari-hari berulang atau berlebihan makan pemampatan tanpa kehilangan dapat dilakukan.

Contoh mudahnya, apabila berkas gambar berukuran 256x256 berwarna polos (setiap pixel berwarna sama) dan tiap pixelnya berukuran 4 byte, tanpa pemadatan, berkas harus disimpan berukuran 4 kali 256x256, sama dengan 262144 byte. Namun, dengan pemadatan, maka data yang perlu disimpan hanyalah data satu warna tersebut dan informasi bahwa seluruh pixel gambar memiliki satu warna yang sama. Jadi, data yang perlu disimpan hanyalah 4 byte tambah beberapa byte untuk menandakan pengulangan pixel yang sama. Ingatlah ini hanya contoh yang simpel.

Pemadatan tanpa kehilangan memiliki batas rendah di mana berkas tidak bisa dipadatkan lebih jauh lagi. Teorem Shannon menunjukkan bahwa pemadatan data tidak bisa menghasilkan kadar kode yang lebih rendah daripada entropi Shannon berkas, tanpa menyebabkan kehilangan informasi. Maka, apabila suatu berkas sudah dipadatkan (misalnya, berkas gambar disimpan di berkas .zip), berkas .zip tersebut tidak bisa lagi dipadatkan.

2. Pemampatan berkehilangan (lossy data compression)

Lossy data compression merupakan kompresi data dengan deviasi, degradasi, atau kehilangan data yang minim dapat diterima dan tidak mengganggu hasil yang akan diterima. Kasus seperti ini dapat ditemukan di audio, video, dan gambar yang degradasi dengan persentase yang minim dapat diterima.

III. DAFTAR ALGORITMA KOMPRESI DATA

A. Lossless data compression

1. Penggunaan Umum

- Run-length encoding (RLE) – Skema sederhana yang dapat memberikan kompresi yang bagus untuk data yang memiliki nilai yang sama.
- Huffman coding – Dapat bekerja dengan baik dengan algoritma lainnya. Digunakan oleh UNIX *pack utility*
- Prediction by partial matching (PPM) – Dioptimasikan untuk plainteks.
- bzip2 – Menggabungkan Burrows–Wheeler transform dengan RLE dan kode Huffman.
- Lempel-Ziv compression (LZ77 and LZ78) – Algoritma berbasis kamus yang menjadi basis algoritma dibawah ini :
    - DEFLATE
    - Lempel–Ziv–Markov chain algorithm
    - Lempel–Ziv–Oberhumer (LZO)
    - Lempel–Ziv–Storer–Szymanski (LZSS)
    - Lempel–Ziv–Welch (LZW)

2. Audio

- Apple Lossless (ALAC - Apple Lossless Audio Codec)
- Adaptive Transform Acoustic Coding (ATRAC)
- apt-X Lossless
- Audio Lossless Coding (dikenal juga sebagai MPEG-4 ALS)
- Direct Stream Transfer (DST)
- Dolby TrueHD
- DTS-HD Master Audio
- Free Lossless Audio Codec (FLAC)
- Meridian Lossless Packing (MLP)
- Monkey's Audio (Monkey's Audio APE)
- MPEG-4 SLS (dikenal juga sebagai HD-AAC)
- OptimFROG
- Original Sound Quality (OSQ)
- RealPlayer (RealAudio Lossless)
- Shorten (SHN)
- TTA (True Audio Lossless)
- WavPack (WavPack lossless)
- WMA Lossless (Windows Media Lossless)

3. Grafis

- PNG – Portable Network Graphics
- TIFF – Tagged Image File Format
- WebP – (densitas yang tinggi dari *lossless* atau *lossy compression* dari RGB dan RGBA)
- BPG – Better Portable Graphics
- FLIF – Free Lossless Image Format
- JPEG-LS – (lossless/near-lossless compression)
- TGA - Truevision TGA
- PCX - PiCture eXchange
- JPEG 2000
- ILBM
- PGF – Progressive Graphics File

4. 3D Grafis

- OpenCTM – Lossless compression of 3D triangle meshes

5. Video
6. Kriptografi
7. Genetika dan Genomika
8. File Eksekusi

B. Lossy data compression

1. Gambar

- Better Portable Graphics **(lossless atau lossy compression)**
- Cartesian Perceptual Compression**,**
- DjVu
- Fractal compression
- ICER**,**
- JBIG2 **(lossless atau lossy compression)**
- JPEG
- JPEG 2000
- JPEG XR
- PGF
- S3TC texture
- Wavelet compression

2. Video

- Motion JPEG
- MPEG-1
- MPEG-2
- MPEG-4
- H.264/MPEG-4 AVC
- Ogg Theora
- Dirac
- Sorenson video codec
- VC-1
- H.265/HEVC

3. Audio

- AAC
- ADPCM
- ATRAC
- Dolby Digital **(AC-3)**
- MP2
- MP3
- Musepack
- Ogg Vorbis
- WMA
- Adaptive Multi-Rate
- Codec2
- Speex

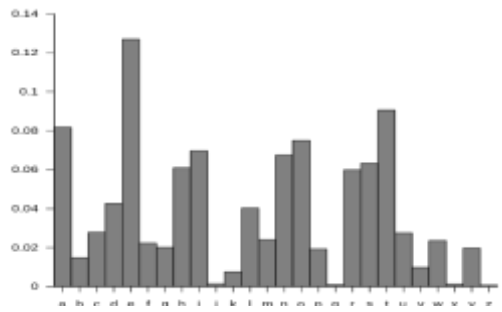
IV. ALGORITMA HUFFMAN

A. Penjelasan Algoritma Huffman

Dalam ilmu komputer dan teori informasi, Huffman coding adalah sebuah tipe code yang optimal yang biasanya digunakan untuk lossless data compression. Algoritma Huffman Coding ditemukan oleh David A. Huffman pada saat ia masih seorang mahasiswa di MIT, ia menerbitkan karyanya di tahun 1952 yang berjudul "*A Method for the Construction of Minimum Redundancy Codes*".

Hasil dari algoritma Huffman bisa dipandang sebagai sebuah tabel kode variabel-panjang untuk pengkodean simbol sumber (seperti sebuah karakter dalam sebuah file). Algoritma ini memperoleh dari tabel tersebut berdasarkan dari estimasi probabilitas atau frekuensi munculnya untuk setiap nilai yang mungkin dari simbol sumber. Seperti dalam metode pengkodean entropi lainnya, simbol yang lebih umum diwakili dengan bit yang lebih sedikit daripada simbol kurang umum.

Di tahun 1951, David A. Huffman dan mahasiswa sekelasnya di teori informasi MIT diberikan pilihan untuk membuat makalah atau mengerjakan ujian akhir. Topik untuk makalah tersebut yang diberikan oleh profesor kelas itu, Robert M. Fano, adalah pencarian kode biner yang paling efisien. Huffman, yang tidak mampu untuk membuktikan kode mana yang paling efisien, hampir menyerah dan sudah mau memutuskan untuk mengambil ujian akhir-nya saja saat tiba-tiba ia terpikir sebuah ide untuk menggunakan algoritma pohon biner yang diurutkan berdasarkan frekuensi. Dengan cepat, ia langsung membuktikan kepada profesornya bahwa metode tersebut adalah metode yang paling efisien.



Gambar 1, ilustrasi tabel 1

Pada gambar 1 diilustrasikan perbandingan frekuensi kemunculan setiap huruf dalam teks berbahasa ingris. Berikut dibawah ini tabel yang lebih rincing yang menyatakan frekuensi relatif huruf alphabet dalam bahasa inggris yang diambil dari buku *Cryptological Mathematics.*

Huruf	Frekuensi Relatif dalam Bahasa Inggris
A	8.167 %
B	1.492%
C	2.782%
D	4.253%
E	12.702%
F	2.228%
G	2.015%
H	6.094%
I	6.966%
J	0.153%
K	0.772%
L	4.025%
M	2.406%
N	6.749%
O	7.507%
P	1.929%
Q	0.095%
R	5.987%
S	6.327%
T	9.056%
U	2.758%
V	0.978%
W	2.360%
X	0.150%
Y	1.974%
Z	0.074%

Tabel 1, Diambil dari buku *Cryptological Mathematics*

B. Rasio Kompresi

Kita dapat menghitung rasio kompresi yang didapatkan menggunakan rumus dibawah ini :

$$Rasio\ Kompresi = \frac{Jumlah\ bit\ setelah\ kompresi}{jumlah\ bit\ sebelum\ kompresi} \cdot 100\%$$

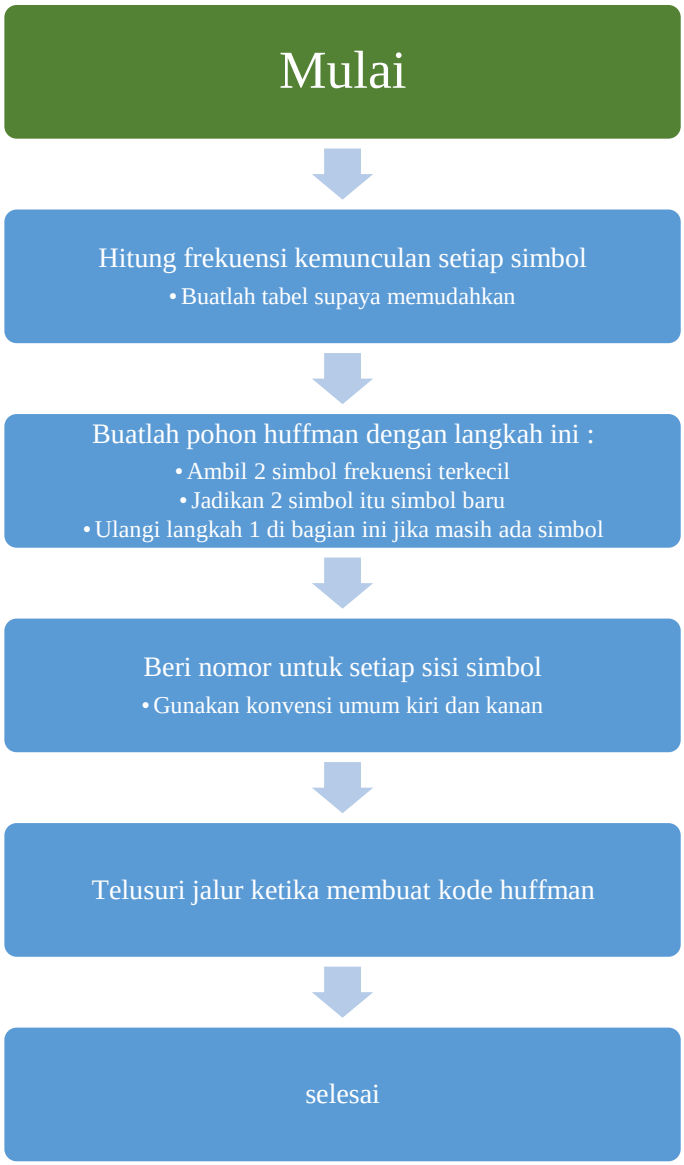
C. Jumlah Bit Setelah Kompresi

Kita dapat menghitung jumlah bit setelah kompresi yang didapatkan menggunakan rumus dibawah ini :

$$\sum_{i=1}^{m} n_i\, f_i$$

Dimana m adalah jumlah simbol unik, n(i) adalah jumlah bit untuk simbol ke I dan f(i) adalah kemunculan simbol ke-i.

Berikut merupakan gambaran pembuatan kode huffman dari menghitung frekuensi sampai pembuatan kode :



D. Pembuatan Pohon Huffman

Algoritma Huffman ini menggunakan pohon biner dalam proses kompresi yang dinamakan pohon huffman. Pohon ini dapat menyimpan data-data dalam bit. Representasi dari data ini bisa apa saja dari mulai huruf, warna display, dan lain-lain. Algoritma untuk membentuk pohon huffman adalah sebagai berikut :

1. Pilih dua data yang memiliki frekuensi paling sedikit. Buatlah dua data ini menjadi daun dari simpul selanjutnya yaitu simpul yang frekuensi kemunculannya merupakan gabungan dari kedua data yang kita pilih sebelumnya.
2. Pilih dua data selanjutnya termasuk simpul yang baru kira buat dan memiliki peluang terkecil.
3. Ulangi langkah 1 dan 2 sampai seluruh data sudah diproses. Pohon yang terbentuk setelah data habis merupakan pohon biner yang sesuai dengan pohon huffman.

E. Pembuatan Kode Huffman

Setelah kita memiliki pohon huffman, kita dapat membuat kode huffman. Berikut beberapa langkah untuk membuat kode huffman dari pohon huffman :

1. Tentukan cabang belah kiri dan kanan
2. Beri nilai untuk cabang kiri dan kanan. Diusahakan pemberian nilai 0 atau 1 sesuai dengan masyarakat atau pembaca sehingga dapat mengerti pohon huffman yang kita buat

F. Contoh Kasus

Disini akan dijelaskan cara membuat pohon huffman dan kode huffman dengan contoh file plainteks.txt yang akan dikompress menjadi huffmancode.bin

Isi Plainteks.txt :

```
Muhamad Arif Adiputra
```

Pertama-tama mari kita buat tabel frekuensi huruf dari seluruh huruf yang muncul diatas

Huruf	Frekuensi Kemunculan
M	2
U	2
H	1
A	5
D	2
R	2
F	1
I	2
P	1
T	1
(Spasi)	2
Total	21

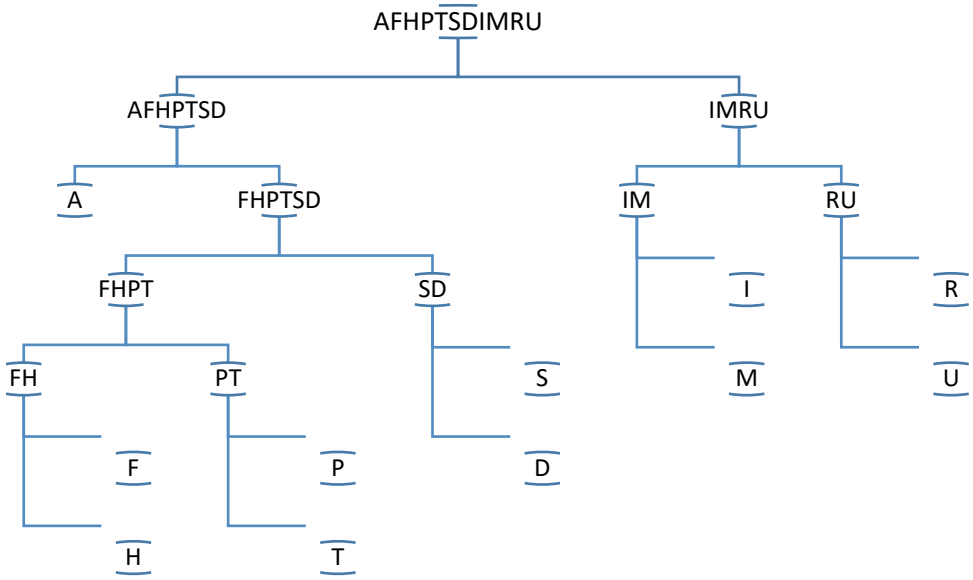
Tabel 2

Pada Tabel 2 dapat terlihat frekuensi kemunculan huruf, mari kita urutkan dari yang terkecil agar mudah menyusun pohon huffman nya.

Huruf	Frekuensi Kemunculan
F	1
H	1
P	1
T	1
(Spasi)	2
D	2
I	2
M	2
R	2
U	2
A	5
Total	21

Tabel 3

Setelah tabel 3 dibuat, mari kita buat pohon biner seperti dibawah ini :



Gambar 2

Setelah pohon huffman dibuat, mari kita buat kode huffman dengan komvensi bagian kiri kode 0 dan kanan 1 (Jika ada yang vertikal di gambar maka yang terdekat nilainya 0 terjauh 1) maka didapatkan kode seperti ini :

Huruf	Kode Huffman
M	101
U	111
H	01001
A	00
D	0111
R	110
F	01000
I	100
P	01010
T	01011
(Spasi)	0110

Tabel 4

Isi huffmancode.bin :

```
10111101001001010011(0110)
0011010001000(0110)
000111000101011101011111000
```

(Dalam Kurung adalah spasi)

Dari sini dapat dilihat bahwa plainteks.txt membutuhkan 21 karakter untuk merepresentasikan data sehingga memakan ruang sebanyak 21 byte (168 bit) sedangkan data dalam huffmancode.txt dengan kode huffman diatas hanya memakan ruang 68 bit.

Perhitungan menggunakan rasio kompresi

$$Rasio\ Kompresi = \frac{Jumlah\ bit\ setelah\ kompresi}{jumlah\ bit\ sebelum\ kompresi} \cdot 100\%$$

$$Rasio\ Kompresi = \frac{68}{168} \cdot 100\%$$

$$Rasio\ Kompresi = 40.47\ \%$$

G. Penerapan Kode Huffman di Gambar Digital

Gambar pada umumnya digunakan hampir di semua bagian dari kehidupan yang menggunakan teknologi. Dari display suatu aplikasi, layar, gambar biasa, kumpulan gambar, sampai video pun menggunakan gambar pada dasarnya untuk merepresentasikan data yang ada. Data yang ada umumnya direpsesentasikan dengan beberapa warna dasar yang dapat membuat warna turunan lainnya salah satu contohnya adalah RGB. Dibawah ini merupakan salah satu contoh gambar yang dikompresi.

Gambar 3, Diambil dari wikipedia.org

Gambar 3 diatas mengalami degradasi kualitas gambar akibat penggunaan *lossy data compression* pada format JPEG. Karena biasanya pada gambar kita tidak memerlukan informasi sedetail mungkin dari gambar maka dapat digunakan kompresi jenis ini agar lebih menghemat ruang. Kompresi huffman dapat digunakan pada gambar hanya kurang besar rasio kompresi nya sehingga tidak terlalu bagus untuk digunakan. Algoritma huffman dipakai di format JPEG juga namun digabung dengan penggunaan jenis *lossy data compression*.

V. KESIMPULAN

Algoritma huffman dapat digunakan untuk plainteks yang mau dikompress dengan baik tanpa kehilangan data sedikitpun dan algoritma ini merupakan aplikasi dari pohon biner. Algoritma huffman bagus digunakan untuk data yang banyak diulang dan perlu digunakan untuk menghemat ruang penyimpanan data

VII. PENGHARGAAN

Alhamdulillah, penulis mengucapkan terima kasih kepada Allah subhanahu wa Ta'ala karena berkat-Nya penulis telah dapat menyelesaikan makalah ini sebagaimana mestinya. Ucapan terima kasih juga saya sampaikan kepada orang tua saya karena dukungan merekalah saya dapat menggenggam kuliah ini.

Terima kasih khususnya kepada dosen pengajar mata kuliah Matematika Diskrit kelas 3 Bapak Dr. Judhi Santoso, M.Sc. yang telah mengajar di kelas kami dan umumnya tim pengajar dosen mata kuliah Matematika Diskrit di kuliah IF semester 3 ini.

REFERENSI

[1] Huffman, D. (1952). "A Method for the Construction of Minimum-Redundancy Codes" (PDF). Proceedings of the IRE. 40 (9): 1098–1101. doi:10.1109/JRPROC.1952.273898. W.-K. Chen, *Linear Networks and Systems* (Book style). Belmont, CA: Wadsworth, 1993, pp. 123–135.

[2] Van Leeuwen, Jan (1976). "On the construction of Huffman trees" (PDF). ICALP: 382–410. Retrieved 20 February 2014. B. Smith, "An approach to graphs of linear forms (Unpublished work style)," unpublished.

[3] Huffman, Ken (1991). "Profile: David A. Huffman: Encoding the "Neatness" of Ones and Zeroes". Scientific American: 54–58.

[4] Gallager, R.G.; van Voorhis, D.C. (1975). "Optimal source codes for geometrically distributed integer alphabets". IEEE Transactions on Information Theory. 21 (2): 228–230. doi:10.1109/TIT.1975.1055357.

[5] Abrahams, J. (1997-06-11). Written at Arlington, VA, USA. Division of Mathematics, Computer & Information Sciences, Office of Naval Research (ONR). "Code and Parse Trees for Lossless Source Encoding". Compression and Complexity of Sequences 1997 Proceedings. Salerno: IEEE: 145–171. CiteSeerX 10.1.1.589.4726 Freely accessible. doi:10.1109/SEQUEN.1997.666911. ISBN 0-8186-8132-2. Retrieved 2016-02-09.

[6] Pujar, J.H.; Kadlaskar, L.M. (May 2010). "A New Lossless Method of Image Compression and Decompression Using Huffman Coding Techniques" (PDF). Journal of Theoretical and Applied Information Technology. 15 (1): 18–23.

[7] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. Introduction to Algorithms, Second Edition. MIT Press and McGraw-Hill, 2001. ISBN 0-262-03293-7. Section 16.3, pp. 385–392.

[8] Huffman, Ken (1991). "Profile: David A. Huffman: Encoding the "Neatness" of Ones and Zeroes". Scientific American: 54–58.

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 4 Desember 2017

Muhamad Arif Adiputra
13516114