

Perbandingan Kinerja Algoritma Huffman dan Algoritma Shannon-Fano Dalam Mengkomperesi File Citra

Krisna Widatama¹, Wahyu Tjahjo Saputro¹

¹Teknologi Informasi, Universitas Muhammadiyah Purworejo, Purworejo 54111, Indonesia

Abstrak

Data yang tersimpan semakin meningkat volumenya. Berbagai algoritma kompresi data banyak diteliti dan dikembangkan. Tujuan penelitian ini adalah untuk mengetahui perbandingan kinerja Algoritma Huffman dan Algoritma Shannon-Fano. Perbandingan kinerja algoritma kompresi dilakukan dengan menguji setiap file citra. Aplikasi yang digunakan untuk menguji dibangun menggunakan Bahasa C++. Empat file berjenis citra akan diuji menggunakan aplikasi yang telah dibangun. File yang diuji mempertimbangan jenis file citra sebagai obyek kompresi. File citra saat ini cenderung meningkat dari sisi ukuran file. Hasil pengujian terhadap empat file citra diperoleh banyak mengandung simbol dan frekuensi dalam menentukan nilai entropi data. File yang sama diuji menggunakan Algoritma Huffman menghasilkan gain kompresi yang lebih baik. File Fish.bmp menggunakan Algoritma Huffman 2.0% dan menggunakan Algoritma Shannon-Fano 1.7%. Semakin besar entropi maka semakin kecil gain yang diperoleh. Misal file putih.bmp entropi: 0.19693 menggunakan Algoritma Huffman di dapat gain 85.2%. file Fish.bmp entropi 6.87332 di dapat gain 2.0%.

Kata Kunci: Kompresi, Citra, Huffman, Shannon-Fano

Abstract

Stored data is increasing in volume. Various data compression algorithms have been researched and developed. The purpose of this study was to compare the performance of the Huffman Algorithm and the Shannon-Fano Algorithm. A comparison of the performance of the compression algorithm is done by testing each image file. The application used to test was built using C++. Four image file types will be tested using the application that has been built. The file being tested considers the image file type as a compression object. The current image file tends to increase in terms of file size. The results of testing the four image files obtained contain many symbols and frequencies in determining the value of data entropy. The same file tested using the Huffman Algorithm produces better compression gain. The Fish.bmp file uses the Huffman 2.0% algorithm and uses the Shannon-Fano algorithm 1.7%. The greater the entropy, the smaller the gain obtained. For example, the white.bmp file entropy: 0.19693 using the Huffman Algorithm can gain 85.2%. entropy Fish.bmp file 6.87332 can get a 2.0% gain.

Keywords: Compression, Image, Huffman, Shannon-Fano

1. PENDAHULUAN

Data yang tersimpan tidak hanya berupa teks, namun juga berupa gambar, suara dan video. Perkembangan teknologi perangkat keras yang menyangkut media penyimpanan juga berkembang pesat, seolah-olah besarnya media penyimpan bukan masalah utama. Dewasa ini perkembangan teknologi media penyimpan cenderung meningkat dengan ditunjukkannya kapasitas media penyimpanan yang semakin besar. Media

penyimpan dengan kapasitas 500GB atau 1TB merupakan hal yang biasa untuk menyimpan sekian banyak data. Namun dengan bertambahnya kapasitas data yang semakin besar perlu dimampatkan atau dikompresi, agar menghemat tempat penyimpanan. Menurut (Umar, 2003), (Kurniawati, 2000) dilihat dari sisi komunikasi data proses kompresi data mempunyai kelebihan tersendiri yaitu, menghemat biaya transmisi data karena data yang dikirim berkapasitas kecil.

Penjelasan (Shanmugam, 1979), (Umar, 2003) pada akhir tahun 1940 dimana dimulainya tahun informasi, ide mengembangkan algoritma kompresi yang efisien baru dimulai. Dimulainya penjelajahan ide dari *entropy*, *information content* dan *redundancy*. Salah satu ide yang populer adalah bila probabilitas dari simbol dalam suatu pesan diketahui, maka terdapat cara untuk mengkodekan simbol, menyebabkan pesan memakan tempat yang lebih kecil. Algoritma pertama yang muncul untuk kompresi data adalah Algoritma Shannon Fano. Shannon dan Fano terus-menerus mengembangkan algoritma yang ditemukan untuk menghasilkan codeword biner untuk setiap simbol bersifat unik pada sebuah data.

Untuk teknologi kompresi data terdapat banyak algoritma diantaranya algoritma Sliding Window, algoritma Huffman, algoritma Shannon Fano, algoritma LZW dan beberapa algoritma lainnya (Umar, 2003), (Kurniawati, 2000), (Simon, 1994), (Shanmugam, 1979), (Simon, 1994). Dalam penelitian ini permasalahan difokuskan pada perbandingan dua algoritma yaitu Huffman dan Shannon Fano. Tujuan penelitian ini yaitu sejauh mana tingkat efisiensi kapasitas file yang telah dikompresi dengan dua algoritma tersebut, guna memilih tingkat kompresi yang baik dan sesuai.

Pada tahun 1952, D. A. Huffman menemukan cara lain untuk kompresi data yang algoritmanya mirip dengan Algoritma Shannon-Fano (Umar, 2003). Tepatnya disebut Algoritma Huffman. Algoritma Huffman ini telah dibuktikan merupakan algoritma yang paling optimal kalau peluang setiap simbol adalah kelipatan $\frac{1}{2}$. Karena tingkat kesulitannya sama, dan memberikan hasil yang lebih baik, Algoritma Huffman menjadi pilihan riset-riset selanjutnya oleh berbagai peneliti dan menjadi Algoritma Kompresi Statistik yang populer ketika itu (Kurniawati, 2000), (Umar, 2003).

Perkembangan selanjutnya terjadi pada tahun 1963, dengan ditemukannya Algoritma Aritmatik (Kurniawati, 2000), (Umar, 2003). Karena algoritma tersebut membutuhkan mesin dengan presisi yang cukup tinggi maka Algoritma Aritmatik menjadi terlalu lambat untuk dapat diimplementasikan secara praktis. Penemuan selanjutnya terjadi pada tahun 1977, dengan ditemukannya algoritma kompresi baru yang menggunakan kamus oleh J. Ziv dan A. Lempel, yang kemudian terkenal dengan Algoritma

LZ77. Tahun 1978 kembali J. Ziv dan A. Lempel mempublikasikan lagi algoritma kompresi sistem kamus yang berbeda. Yang dikenal dengan sebutan Algoritma LZ78 (Umar, 2003).

Namun Algoritma LZ78 mulai dikenal tahun 1982, setelah Stoner dan Szymanski memodifikasi agar lebih mudah digunakan. Modifikasi ini dikenal dengan sebutan LZSS. Tetapi popularitasnya tidak terlalu lama, karena pada tahun 1984, Terry Welch memodifikasi LZ78, agar lebih mudah diimplementasikan dalam program komputer mikro. Dan dikenal dengan Algoritma LZW. Rasio kompresi Algoritma LZW lebih baik dari Algoritma LZSS yang mengakibatkan banyak orang menggunakan Algoritma LZW. Tampaknya LZ77 akan tamat riwayatnya, tetapi dengan munculnya paten LZW dan ditemukannya berbagai cara untuk dapat meningkatkan kemampuan LZ77, justru lebih populer dari LZ78 dan turunannya (Umar, 2003).

Tahun 1985 ditandai dengan munculnya berbagai teknik model baru untuk kembali menjalankan Algoritma Statistik yang lebih cepat, juga disusul dengan berkembangnya teori pendekatan yang digunakan untuk mengembangkan algoritma kompresi *lossy* (Kurniawati, 2000).

Menurut (Umar, 2003) kompresi data adalah usaha untuk mencari dan menemukan jumlah bit yang lebih sedikit untuk menyimpan atau mengirim informasi. Kompresi data dapat dibagi menjadi dua yaitu :

1. *Lossy Compression* (kompresi lemah)
2. *Lossless Compression* (kompresi kuat)

Kompresi lemah mengijinkan adanya sejumlah kehilangan ketelitian dengan batas tertentu. Sebagai gantinya adalah penambahan resiko kompresi yang besar. Kompresi lemah biasa digunakan untuk mengkompresi file citra dan suara. Sedangkan kompresi kuat yaitu bila data sebelum dikompres dan setelah dikompres akan sama persis. Sebab bila terjadi sedikit kesalahan, meskipun hanya satu bit dapat berakibat kesalahan fatal. Kompresi kuat biasa digunakan untuk file basis data, biner dan teks.

Algoritma Huffman memakai hampir semua karakteristik dari Shannon-Fano. Algoritma Huffman dapat menghasilkan kompresi data yang efektif dengan mengurangi jumlah redundansi dalam mengkodekan simbol. Telah dibuktikan bahwa Algoritma Huffman merupakan Teknik *Fixed-Length* yang paling efisien. Pada

15 tahun terakhir Algoritma Huffman telah digantikan oleh Metode Aritmatika (Umar, 2003). Metode Aritmatika melewatkan ide untuk menggantikan sebuah simbol masukan dengan sebuah angka keluaran *single floating-point*. Bila lebih banyak bit dibutuhkan dalam angka keluaran, maka semakin rumit pesan yang diterima.

Pandangan dasar dari kode simbol adalah menghilangkan redundansi dari sumber. Kode simbol menghasilkan data kompresi dan mengurangi rate dari transmisi. Pengurangan dari rate transmisi dapat mengurangi biaya dari sambungan dan memberikan kemungkinan untuk pengguna berbagi dalam sambungan yang sama. Secara umum dapat di kompresi data tanpa menghilangkan informasi (*lossless source coding*). Teori *Source Encoding* merupakan salah satu dari ketiga teori dasar dari teori informasi yang dikenalkan oleh Shannon. Teori *Source Encoding* mencanangkan sebuah limit dasar dari sebuah ukuran dimana keluaran dari sebuah sumber informasi dapat di kompresi tanpa menyebabkan probabilitas error yang besar. Telah diketahui bersama bahwa entropi dari sebuah sumber informasi merupakan sebuah ukuran dari isi informasi pada sebuah sumber. Maka dari pendapat Teori *Source Encoding* bahwa entropi dari sebuah sumber sangat penting (Kurniawati, 2000).

Pemampatan data pada kompresi data *lossless* dilakukan tanpa menghilangkan bagian dari informasi. Kompresi data jenis ini harus mampu mengembalikan kode hasil kompresi ke dalam bentuk urutan simbol yang persisi sama seperti aslinya. Dalam bahasa matematik, jika W adalah data dan $F()$ adalah fungsi maka untuk memampatkan datanya adalah (Simon, 1994), (Shanmugam, 1979) :

$$F^{-1}(F(W)) = W$$

Metode *lossless* ini digunakan untuk berbagai kompresi data yang tidak boleh berubah sedikit pun, seperti telah dikemukakan di awal. Misal data teks (jika berubah isinya maka artinya pun berubah. Namun Metode *Lossless* mempunyai keterbatasan, yaitu kemampuan kompresinya rendah. Untuk data suara dan gambar, kemampuan kompresinya sekitar 2:1 atau 3:1 (Umar, 2003).

Kekurangan Metode *Lossless* ditanggulangi dengan menggunakan Metode *Lossy*. Metode

Lossy dilakukan dengan menghilangkan bagian informasi yang dianggap tidak penting maka dihalikan rasio kompresi yang sangat tinggi, dengan demikian data hasil pemekaran tidak sama dengan aslinya. Dampaknya adalah output dari Metode *Lossy* cenderung menjadi lebih halus atau lebih rata. Dalam bahasa matematik jika W adalah data dan $F()$ adalah fungsi untuk memampatkan data maka (Shanmugam, 1979), (Simon, 1994):

$$F^{-1}(F(W)) \neq W$$

Tetapi dapat di buat sedekat mungkin dengan aslinya, dengan mengatur parameter kualitas (δ), maka dalam bahasa matematis di dapat formula berikut (Shanmugam, 1979), (Simon, 1994):

$$|W - F^{-1}(F(W))| < \delta$$

Dengan nilai δ dapat dibuat sekecil mungkin. Bila $\delta = 0$, maka Metode *Lossy* menjadi Metode *Lossless*. Kelebihan Metode *Lossy* adalah kemampuan kompresi datanya sangat besar, dapat mencapai 100:1 atau lebih dan dapat diatur. Jika kompresi besar maka δ juga besar, tetapi jika kompresi kecil maka δ juga kecil (Kurniawati, 2000). Ini memberikan fleksibilitas untuk kompresi data-data yang akan dihadapi oleh pengguna. Misalnya data gambar dan suara. Fakta bahwa indera manusia memiliki keterbatasan mendukung untuk menghilangkan bagian dari informasi yang tidak peka terhadap persepsi visual dan pendengaran manusia.

2. METODE

2.1. Bahan Yang Digunakan

Bahan yang digunakan dalam penelitian ini adalah program kompresi data yang dibuat dengan Bahasa C++ (Kadir, 1995), (Weiskamp, 1997). Program ini digunakan untuk menguji kompresi data dengan Algoritma Huffman dan Shannon-Fano. File yang diuji adalah file citra bitmap (.BMP). Ada lima file yang diuji dengan rincian pada Tabel 1.

Tabel 1. Daftar File Yang Akan Dikompresi

Nama File	Jenis File	Ukuran
-----------	------------	--------

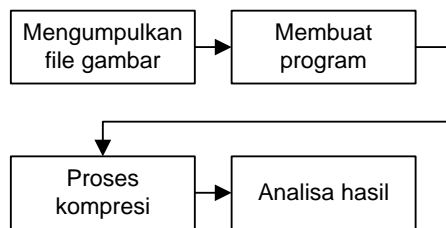
Putih.bmp	Citra	6Kb
Gambar.bmp	Citra	4Kb
Stone.bmp	Citra	12Kb
Fish.bmp	Citra	17Kb

2.2. Perangkat Yang Digunakan

Program yang digunakan dibuat menggunakan Borland C++ 5.02, dengan sistem operasi Windows. Spesifikasi komputer yang digunakan adalah: Intel Core 2 Duo 2.9GHz P3900, RAM 4MB.

2.3. Cara Penelitian

Cara yang digunakan dalam penelitian ini menggunakan Metode Eksperimen, yaitu melakukan pengujian terhadap keempat file gambar yang telah disiapkan, kemudian di kompres dengan aplikasi yang telah dibuat. Program akan menjalankan satu-persatu dimana setiap proses akan mengkompresi file dari jenis dan ukuran yang sama dan menggunakan spesifikasi komputer sama. Tahap penelitian ditunjukkan Gambar 1.



Gambar 1. Tahap Penelitian

3. HASIL DAN PEMBAHASAN

3.1. Algoritma Huffman

Menurut Huffman dalam (Umar, 2003) dan (Kurniawati, 2000), metode kompresi menggunakan bermacam-macam panjang kode untuk mengungkapkan karakter-karakter. Kode Huffman mengalokasikan kode yang paling pendek untuk simbol dengan probabilitas paling tinggi. Dan kode yang panjang untuk simbol dengan probabilitas yang rendah. Implementasi Algoritma Huffman dalam program biasanya menggunakan pohon biner yang terdiri atas simpul-simpul. Simpul ini mengandung informasi simbol yang diwakili probabilitasnya. Simpul yang lebih dekat ke akar, mendapat panjang kode lebih pendek dibanding simpul yang terletak jauh dari akar.

Suatu simpul akan dipasang sebagai cabang kanan jika probabilitasnya lebih besar dari simpul induk dan akan diletakkan di cabang kiri jika nilai probabilitasnya lebih kecil dari simpul induk. Algoritma Huffman dan flowchart dengan menggunakan pohon biner sebagai berikut:

1. Pengurutan keluaran sumber di mulai dari probabilitas paling tinggi.
2. Menggabungkan dua keluaran yang sama dekat ke dalam satu keluaran yang probabilitasnya merupakan jumlah dari probabilitas sebelumnya.
3. Apabila setelah di bagi msaih terdapat dua keluaran, maka lanjutkan ke langkah berikutnya, namun bila masih terdapat lebih dari dua, maka kembali ke langkah pertama.
4. Memberikan nilai 0 dan 1 untuk kedua keluaran.
5. Apabila sebuah keluaran merupakan hasil dari penggabungan dua keluaran dari langkah sebelumnya, maka berikan tanda 0 dan 1 untuk kode simbolnya. Ulangi sampai keluaran merupakan satu keluaran yang berdiri sendiri.

Sebagai ilustrasi diambil kata: “HUFFMAN”. Langkah pertama melakukan kalkulasi probabilitas dari setiap simbol. Hasilnya pada Tabel 2.

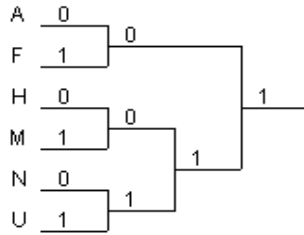
Tabel 2. Hasil probabilitas setiap simbol

Simbol	Probabilitas
A	1/7
F	2/7
H	1/7
M	1/7
N	1/7
U	1/7

Selanjutnya proses Huffman Tree dari kalimat “H U F F M A N” langkahnya seperti Gambar 2.

3.2. Algoritma Shannon-Fano

Algoritma Shannon-Fano didasarkan pada variabel *length-word*, yang berarti beberapa simbol pada data direpresentasikan dengan kode simbol yang lebih pendek dari simbol yang ada dalam data. Semakin tinggi probabilitasnya, maka kode simbol semakin pendek.



Gambar 2. Huffman Tree

Dalam memperkirakan panjang setiap kode simbol tersebut, algoritma Shannon-Fano menghasilkan kode simbol yang tidak sama panjang, maka kode tersebut bersifat unik dan dapat didekodekan.

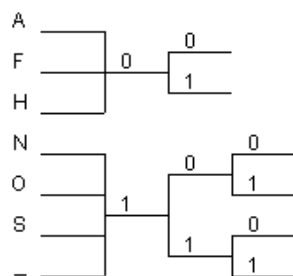
Cara efisien lainnya dalam variable *length-coding* adalah Shannon-Fano *encoding*. Prosedur dalam Shannon-Fano *encoding* adalah:

1. Menyusun probabilitas simbol dari sumber yang paling tinggi ke yang paling rendah.
2. Membagi menjadi dua bagian yang sama besar, dan memberikan nilai 0 untuk bagian atas dan 1 untuk bagian bawah.
3. Ulangi langkah ke dua, setiap pembagian dengan probabilitas yang sama sampai dengan tidak mungkin dibagi lagi.
4. Encoding setiap simbol asli dari sumber menjadi urutan biner yang dibangkitkan oleh setiap proses pembagian tersebut.

Sebagai ilustrasi diambil kata “SHANNON-FANO”. Langkah pertama melakukan kalkulasi dari sebuah simbol. Hasilnya seperti Tabel 3. Kemudian struktur pohon seperti Gambar 3.

Tabel 3. Hasil probabilitas setiap simbol

Simbol	Probabilitas
A	2/12
F	1/12
H	1/12
N	4/12
O	2/12
S	1/12
-	1/12



Gambar 3. Shannon-Fano Tree



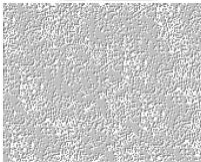

3.3. Permasalahan Yang Dianalisa

Untuk melakukan simulasi program kompresi data di matlab, di sini hanya menggunakan file citra bitmap (bmp) dengan ukuran yang berbeda satu sama lain. Analisa proses kompresi pada penelitian ini hanya akan menjelaskan kompresi citra. Permasalahan yang akan di analisa yaitu :

1. Pengaruh banyaknya simbol terhadap gain kompresi.
2. Pengaruh banyaknya frekuensi tiap simbol terhadap gain kompresi.
3. Pengaruh nilai entropi terhadap gain kompresi.
4. Kelebihan dari masing-masing metode kompresi yang digunakan dalam program simulasi.
5. Kekurangan dari masing-masing metode kompresi yang digunakan dalam program simulasi.

File citra yang dijadikan sebagai data ada empat, seperti tercantum dalam Tabel 4.

Tabel 4. Daftar file yang diuji

Bentuk Citra	Nama File
	Putih.bmp (6 kb)
	Gambar.bmp (4 kb)
	Stone.bmp (12 kb)
	Fish.bmp (17 kb)

Tabel 6. Hasil kompresi dengan Algoritma Huffman

Nama File	Entropi	Panjang Kode (Sebelum)	Panjang Kode (Sesudah)	Gain (%)
Putih.bmp	0.19693	42700 bits	6318 bits	85.2037
Gambar.bmp	2.40134	25200 bits	8799 bits	65.0833
Stone.bmp	3.15378	70000 bits	31914 bits	54.4086
Fish.bmp	6.87332	70000 bits	68599 bits	2.00143

Tabel 7. Hasil kompresi dengan Algoritma Shannon-Fano

Nama File	Entropi	Panjang Kode (Sebelum)	Panjang Kode (Sesudah)	Gain (%)
Putih.bmp	0.19693	42700 bits	6318 bits	85.2037
Gambar.bmp	2.40134	25200 bits	8816 bits	65.0159
Stone.bmp	3.15378	70000 bits	31977 bits	54.3186
Fish.bmp	6.87332	70000 bits	68770 bits	1.75714

Dari Tabel 4 di atas diketahui bahwa kompleksitas setiap gambar berbeda. Putih.bmp mempunyai kompleksitas paling rendah dan fish.bmp mempunyai kompleksitas paling tinggi. Kompleksitas gambar ditentukan oleh banyaknya gradasi warna (*gray level*) setiap gambar. Banyaknya *gray level* menentukan banyaknya simbol yang harus dikodekan. Gambar warna hitam-putih (*grayscale*) dikodekan oleh 8 bit. Artinya akan ada 2^8 level warna hitam. Warna paling hitam disimbolkan dengan 0, sedangkan warna paling putih disimbolkan dengan 255. Tabel 5 berikut adalah banyaknya *gray level* dari setiap gambar pada Tabel 4.

Tabel 5. Banyaknya *gray level* pada setiap file

Nama File	Banyaknya Gray Level
Putih.bmp	15
Gambar.bmp	72
Stone.bmp	60
Fish.bmp	237

Kemudian Tabel 6 dan 7 adalah hasil dari kompresi yang telah dilakukan menggunakan program yang dibuat dari bahasa C++.

Pada kedua algoritma baik Huffman maupun Shannon-Fano file putih.bmp memiliki entropi paling kecil yaitu 0.19693. Berarti nilai informasinya sangat kecil. Terbukti bahwa hampir semua simbol putih.bmp adalah 255 (*gray level* 255) yang jumlahnya 5940. Jumlah simbol-simbol lain tidak terlalu besar. Simbol dengan probabilitas terbesar dikodekan hanya dengan 1 bit. Berarti semua *gray level* 255 dikodekan dengan 1 bit. Simbol-simbol dengan probabilitas kecil, dikodekan dengan 6 bit. Sebelum

dikodekan dengan kode Huffman, setiap simbol dikodekan dengan 8 bit. Karena itulah gain kompresi yang di dapat sangat tinggi sekitar 85.2%.

Pada file fish.bmp untuk dua algoritma mempunyai nilai entropi yang tinggi yaitu 6.87%. Ini dapat ditunjukkan dari banyaknya simbol dengan frekuensi yang hampir merata di setiap simbol. Karena itulah fish.bmp hanya dapat di kompresi dengan gain 2% dengan algoritma Huffman. Bila diperhatikan, gambar.bmp memiliki simbol lebih banyak dari pada stone.bmp. Namun setelah di kompresi justru gambar.bmp yang paling kecil hasilnya, lihat Tabel 6 dan 7 pada kolom panjang kode sesudah di kompresi. Karena frekuensi dari setiap simbol pada gambar.bmp kurang merata dibandingkan stone.bmp. Pada file stone.bmp frekuensi dari setiap simbol hampir sama, lihat Tabel 4.

Dari segi lamanya proses pengkodean, algoritma Huffman membutuhkan waktu lebih lama dari Shannon-Fano. Ini disebabkan Algoritma Huffman yang lebih panjang dan kompleks dari pada Algoritma Shannon-Fano. Pada algoritma Huffman, simbol-simbol harus diurutkan dari yang paling tinggi probabilitasnya sampai yang paling rendah. Kemudian dua probabilitas terkecil dijumlahkan dan hasilnya diurutkan kembali dari yang paling besar sampai yang paling kecil. Begitu seterusnya sampai akhir penjumlahannya bernilai 1.

Sedangkan pada Shannon-Fano, simbol-simbol hanya satu kali diurutkan dari yang probabilitasnya paling kecil sampai yang paling besar. Kemudian dari urutan simbol-simbol tersebut

dibagi dua kelompok berdasarkan probabilitasnya. Kemudian hasilnya dibagi dua kelompok berdasarkan probabilitasnya. Dan hasilnya dibagi dua kelompok lagi. Begitu seterusnya sampai setiap kelompok tidak dapat lagi dibagi dua dan hanya tersisa satu simbol.

3.4. Cara Kerja Kompresi Data

Bagaimana cara komputer membaca dan membedakan deretan bit yang sudah dikodekan ?. Di ambil contoh masukan dari user berupa teks “maulana”. File ini di simpan lalu dikodekan dengan algoritma Huffman, hasilnya seperti Tabel 8.

Tabel 8. Hasil probabilitas setiap simbol

Simbol	Probabilitas	Kode Huffman
a	3/7	0
m	1/7	1 0 0
u	1/7	1 0 1
l	1/7	1 1 0
n	1/7	1 1 1

Pada saat data di kirim, urutan bitnya menjadi:

1 0 0 0 1 0 1 1 1 0 0 1 1 1 0

Dari urutan bit ini, komputer akan membaca bit terdepan (paling kiri) sampai terakhir (paling kanan). Selain membaca bit, komputer juga akan membandingkan dengan tabel kode yang juga dikirimkan bersama data. Apabila dalam pembacaan urutan bit kemudian dibandingkan dengan tabel ada kecocokan, maka komputer langsung menterjemahkan ke dalam simbol aslinya. Ini yang di sebut unique prefix. Contoh:

- Iterasi 1** Komputer membaca bit **1**: Komputer mencari di tabel apakah simbol untuk kode **1**, ternyata tidak ada.
- Iterasi 2** Komputer membaca bit **1 0**: Komputer mencari di tabel apakah simbol untuk kode **1 0**, ternyata tidak ada.
- Iterasi 3** Komputer membaca bit **1 0 0**: Komputer mencari di tabel apakah simbol untuk kode **1 0 0**, ternyata simbol **m**.
- Iterasi 4** Komputer membaca bit **0**: Komputer mencari di tabel apakah simbol untuk kode **0**, ternyata simbol **a**.

Dan seterusnya sampai semua kode berhasil diterjemahkan ke simbolnya masing-masing.

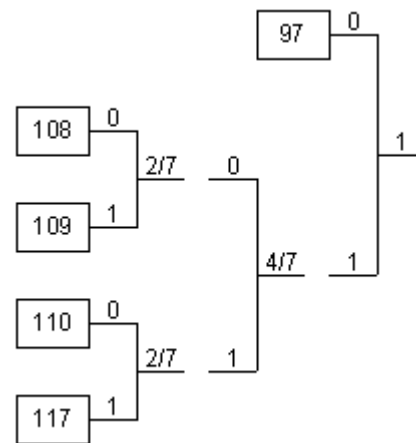
3.5. Verifikasi Program

Untuk verifikasi kebenaran algoritma program, dimasukkan data dari user.txt yang isinya berupa teks: **maulana**. User.txt kemudian dibaca oleh program dan hasilnya pada Tabel 9.

Tabel 9. Hasil verifikasi data

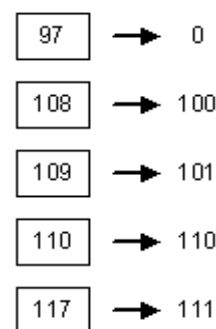
Simbol	97	198	109	110	117
Frekuensi	3	1	1	1	1

Artinya simbol **a** dinyatakan dengan kode ASCII 1011101 (biner dari 97), begitu juga simbol-simbol yang lain dibaca dengan kode ASCII. Dengan Metode Huffman maka jalannya proses pengkodean seperti Gambar 4.



Gambar 4. Pengkodean algoritma Huffman

Hasil kode Huffman dari simbol-simbol gambar 4 adalah tampak pada Gambar 5. Kode biner tersebut diperoleh dari menarik angka 1 dan 0 dari sisi kanan ke kiri.



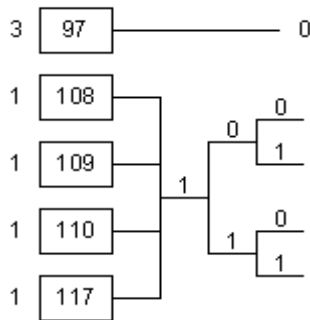
Gambar 5. Hasil pengkodean Huffman dari Gambar 4

Berikut adalah perhitungan entropi dari masing-masing simbol :

- Simbol **a** probabilitas 3/7 maka besar entropi : $(-3/7)^2 \log 3/7 = 0.5239$

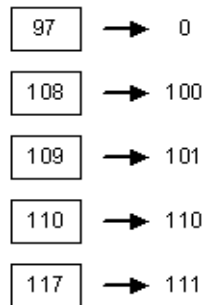
2. Simbol **m** probabilitas 1/7 maka besar entropi: $(-1/7)^2 \log 1/7 = 0.4011$
3. Simbol **u** probabilitas 1/7 maka besar entropi : $(-1/7)^2 \log 1/7 = 0.4011$
4. Simbol **l** probabilitas 1/7 maka besar entropi : $(-1/7)^2 \log 1/7 = 0.4011$
5. Simbol **n** probabilitas 1/7 maka besar entropi : $(-1/7)^2 \log 1/7 = 0.4011$

Maka jumlah total entropi adalah: 2.1281. Dengan demikian entropi memakai program dan manual adalah sama. Kemudian algoritma yang kedua adalah Shannon-Fano, jalannya proses pengkodean adalah pada Gambar 6.



Gambar 6. Proses pengkodean Shannon-fano

Hasil kode Shannon-Fano dari simbol-simbol Gambar 6 tampak pada Gambar 7.



Gambar 7. Hasil pengkodean Shannon-Fano dari Gambar 6

Dari Gambar 6 dan 7, tampak bahwa proses entropi dengan menggunakan cara manual dan proses olahan program hasilnya sama.

4. KESIMPULAN

Banyaknya simbol dan frekuensi setiap simbol menentukan nilai entropi suatu data. Nilai entropi

menentukan gain kompresi. Makin besar entropi maka akan semakin kecil gain yang diperoleh pada saat pengkompresan suatu data. Dengan file yang sama, algoritma Huffman menghasilkan gain kompresi yang lebih baik dibandingkan Algoritma Shannon-Fano, lihat tabel 6 dan 7. Hasil kompresi akan dikirimkan ke tujuan berikut dengan tabel kodenya untuk proses dekoding. Urutan biner hasil koding tidak akan salah dikodekan karena setiap kode memiliki unique prefix. Pada file citra dengan ukuran sama dengan banyak warna yang digunakan sangat mempengaruhi rasio hasil kompresi. Semakin sedikit warna yang digunakan maka semakin besar rasio hasil kompresi diperoleh jika dibanding dengan file yang menggunakan banyak warna.

DAFTAR PUSTAKA

Kadir, A. (1995). *Pemrograman C++*. Yogyakarta, DIY, Indonesia: Andi Publisher.

Kurniawati, D. (2000). *Perbandingan Kinerja Algoritma Kompresi Data Metode Huffman dan Metode Sliding-Window*. Teknik Informatika. Yogyakarta: STMIK AKAKOM.

Shanmugam, S. K. (1979). *Digital and Analog Communications System*. Hoboken, New Jersey, Amerika: John Wiley and Sons, Inc.

Simon, H. (1994). *An Introduction to Analog and Digital Communication*. Hoboken, New Jersey, Amerika: John Wiley Inc.

Umar, R. (2003, Desember). Unjuk Kerja Algoritma LZW Untuk Kompresi Berkas Teks. *Telkomnika*, 1(1), 7-14.

Weiskamp, K. (1997). *Borland C++ 4.0 Premier* (Indonesia ed., Vol. Cetakan Kedua). Jakarta, DKI, Indonesia: Dinastindo.