Aritmathic Code Data Compression In Building Data Communication

Faizal *[1], Rachmat[2], Abdul Ibrahim[3]
[1,3]Universitas Dipa Makassar, [2]Politeknik LP3I Makassar
[1,3]Jl. Perintis Kemerdekaan Km.9, [2]JL. Sultan Alauddin No.250.
e-mail:*[1]ichalabinurullah@gmail.com,[2]rachmat27udinus@gmail.com,
[3]abdulibrahim@dipanegara.ac.id,

Abstrak

Output dari sumber-sumber ini adalah sinyal analog dan sumbernya disebut sumber analog. Kontras dengan komputer dan penyimpanan seperti disk magnetik atau optik, menghasilkan sinyal diskrit (biasanya karakter biner atau ASCII). Sistem kompresi data dalam bentuk sinyal analog ini menarik untuk dikaji dalam pensinyalan analog. Dari penelitian ini, penulis membuat penelitian tentang kompresi data dalam jaringan Komputer, Algoritma kompresi berbasis kamus menggunakan metode yang sangat berbeda untuk mengompresi data. Algoritma ini menggantikan string panjang variabel dari sebuah simbol menjadi sebuah token. Pengkodeaan aritmatika dapat memanfaatkan ide untuk menggantikan sebuah simbol masukan dengan kode yang spesifik. Algoritma ini menggantikan sebuah aliran simbol masukan dengan sebuah angka keluaran tunggal. Lebih banyak bit dibutuhkan dalam angka keluaran, maka semakin rumit pesan yang diterima. Pengodean Aritmatika dapat menggantikan string simbol input dengan angka floating point. Semakin lama dan semakin kompleks pesan yang dikodekan, semakin banyak bit yang dibutuhkan untuk tujuan itu. hasil dari pengkodean aritmatika ini adalah satu angka kurang dari 1 dan lebih besar dari atau sama dengan 0. Penomoran ini dapat didekodekan secara unik, untuk menghasilkan sekumpulan simbol yang digunakan dalam menghasilkan penomoran. Untuk dapat menghasilkan nilai keluaran, setiap simbol yang akan dikodekan diberi satu set nilai probabilitas.

***Kata kunci**: sinyal analog, komputer, kompresi data, kode aritmatika*

Abstract

The output of these sources is an analog signal and the source is called an analog source. In contrast to computers and storage such as magnetic or optical disks, they produce discrete signals (usually binary or ASCII characters). This data compression system in the form of an analog signal is interesting to study in analog signaling. From this research, the writer makes a research about data compression in computer network, the dictionary based compression algorithm uses a very different method to compress the data. This algorithm replaces a variable length string of a symbol into a token. Arithmetic coding can take advantage of the idea of replacing an input symbol with a specific code. This algorithm replaces a stream of input symbols with a single output number. The more bits needed in the output number, the more complicated the message will be. Arithmetic encoding can replace input symbol strings with floating point numbers. The longer and more complex the encoded message, the more bits it will need for that purpose. the result of this arithmetic encoding is one number less than 1 and greater than or equal to 0. This numbering can be uniquely decoded, to produce a set of symbols used in generating the numbering. To be able to produce an output value, each symbol to be encoded is assigned a set of probability values.

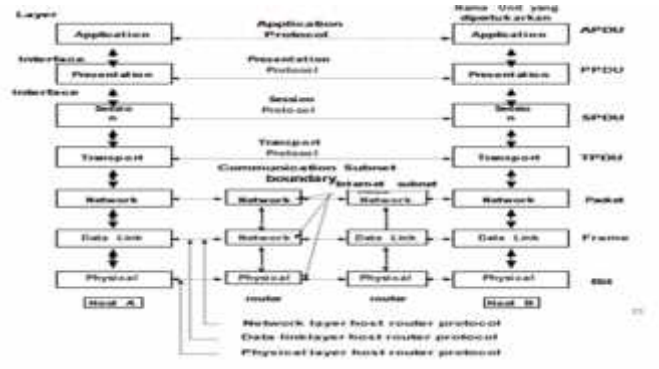
Keywords: *analog signal, computer, data compression, arithmetic code*

1. PENDAHULUAN

Sistem komunikasi dirancang untuk mengirimkan informasi yang dihasilkan oleh sumber ke berbagai tujuan. Sumber informasi memiliki beberapa bentuk yang berbeda. Misalnya, dalam siaran radio, sumber biasanya berupa sumber audio (suara atau musik).[1][2] Output dari sumber-sumber ini adalah sinyal analog dan sumbernya disebut sumber analog mirip dengan sistem komputer dan penyimpanan seperti disk magnetik atau optik, menghasilkan sinyal diskrit (biasanya karakter biner atau ASCII).[3] Sinyal yang menjadi sumber biasanya disebut sebagai sumber diskrit. baik analog dan sumber diskrit, komunikasi digital dirancang untuk mengirimkan informasi dalam bentuk digital. Jadi kesimpulan keluaran dari sumber harus diubah terlebih dahulu menjadi bentuk keluaran sumber digital yang biasanya dilakukan pada source encoder.[4] Output dari sumber dapat diasumsikan sebagai digit biner berurutan.[5] Pada akhir tahun 40-an, ketika tahun teori informasi dimulai, gagasan untuk mengembangkan metode pengkodean yang efisien dimulai dan dikembangkan. Awal dari eksplorasi ide mulai dari entropi, kandungan informasi dan redudansi.[6][7] Salah satu ide yang populer adalah jika probabilitas sebuah simbol dalam sebuah pesan diketahui, maka ada cara untuk mengkodekan simbol tersebut, sehingga pesan tersebut menempati ruang yang lebih kecil. Ide inilah yang menjadi cikal bakal dalam pembuatan kompresi data. Model pertama yang muncul untuk kompresi sinyal digital adalah pengkodean Shannon-Fano. Shannon dan fano (1948) terus mengembangkan algoritma ini yang menghasilkan kode biner untuk setiap simbol yang terkandung dalam file data.[8][9] Pengkodean Huffman (1952)[10] memakai hampir semua karakteristik pengkodean Shannon-fano. Metode pengkodean Huffman dapat menghasilkan kompresi data yang efektif dengan mengurangi jumlah redundansi sinyal analog dalam simbol pengkodean. Telah terbukti bahwa pengkodean Huffman adalah metode panjang tetap yang paling efisien. Dalam lima belas tahun terakhir, pengkodean Huffman telah digantikan oleh pengkodean Aritmatika.[11] Pengkodean aritmatika melewatkan gagasan untuk mengganti simbol input dengan kode tertentu. Algoritma ini menggantikan aliran simbol input dengan nomor output floating-point tunggal. Semakin banyak bit yang dibutuhkan dalam nomor keluaran, semakin rumit pesan yang diterima.*Arithmetic coding* memanfaatkan ide untuk menggantikan sebuah simbol masukan dengan kode yang spesifik. Algoritma ini menggantikan sebuah aliran simbol masukan dengan sebuah angka keluaran *single floating-point*. Lebih banyak bit dibutuhkan dalam angka keluaran, maka semakin rumit pesan yang diterima. Kompresi data (pemanfaatan data) merupakan suatu teknik untuk memperkecil jumlah ukuran data (hasil kompresi) dari data aslinya. Pemampatan data umumnya diterapkan pada mesin komputer, hal ini dilakukan karena setiap simbol yang muncul pada komputer memiliki nilai bit-bit yang berbeda.

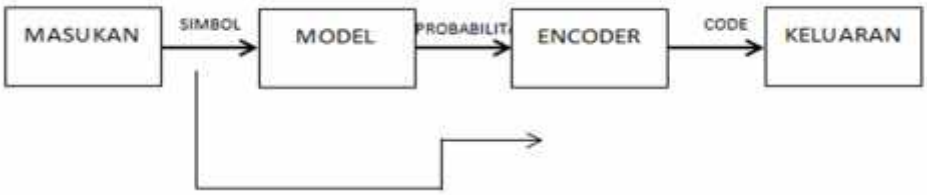
2. METODE PENELITIAN

Secara umum, kompresi data adalah mengubah simbol menjadi kode. Kompresi dikatakan efektif jika ukuran yang diperoleh dari kode-kode tersebut sangat kecil dibandingkan dengan ukuran kode simbol aslinya. Dari sebuah kode atau simbol dasar sebuah model akan diekspresikan dalam kode khusus.[13][14]

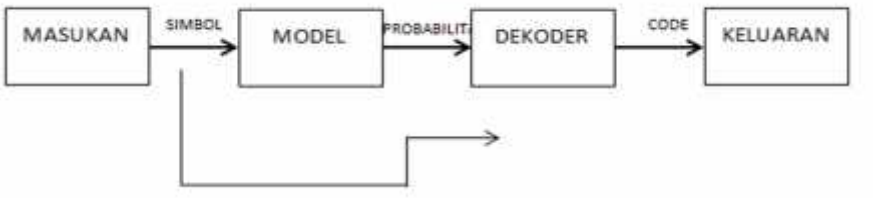


Gambar 1. Model Referensi OSI

Proses decoding, yaitu proses mengembalikan kode-kode yang telah dibuat menjadi simbol-simbol yang diketahui oleh pengguna. Proses decoder ini membaca header dari kode-kode yang berisi informasi simbol dan jumlah simbol yang digunakan, setelah membaca header proses encoder dengan model, sedangkan proses decoder dengan model.



Gambar 2 Encoder Process Model



Gambar 3. Decoder Process Model

3.1 Pengkodean Data

Data disimpan di komputer pada memori utama untuk diproses. Sebuah karakter data yang disimpan dalam memori utama menempati posisi 1 byte. Komputer generasi pertama, 1 byte terdiri dari 4 bit, komputer generasi kedua, 1 byte terdiri dari 6 bit dan komputer generasi sekarang, 1 byte terdiri dari 8 bit. Suatu karakter data yang disimpan dalam memori utama direpresentasikan dengan kombinasi angka biner (binary digit atau bits). Sebuah kode biner dapat digunakan untuk mewakili sebuah karakter.[13] Komputer yang berbeda menggunakan kode biner yang berbeda untuk mewakili karakter. Komputer 1 byte terdiri dari 4 bit, menggunakan kode biner berupa kombinasi 4 bit yaitu BCD (Binary coded decimal). Komputer yang menggunakan 6 bit untuk 1 byte, menggunakan kode biner yang terdiri dari kombinasi 6 bit, yaitu SBCDIC (Standard Binary Coded Decimal Interchange Code).[15] Komputer yang terdiri dari 8 bit, menggunakan kode biner yang terdiri dari kombinasi 8 bit, yaitu kode EBCDIC (extended Binary coded decimal interchange code) atau ASCII (American standard code of information interchange). (Zhu dkk. 1999)

3.2 BCD (Desimal Berkode Biner)

BCD adalah kode biner yang digunakan untuk mewakili nilai desimal saja, yaitu angka 0 sampai dengan 9. BCD menggunakan kombinasi 4 bit, sehingga dapat diperoleh sebanyak 16 (24 = 16) kemungkinan kombinasi dan hanya 10 kombinasi yang digunakan.

Kode BCD asli jarang digunakan untuk komputer generasi saat ini, karena tidak dapat mewakili huruf atau simbol karakter khusus.

3.3 SBCDIC (Kode Pertukaran Desimal Berkode Biner Standar)

SBCDIC adalah kode biner perkembangan BCD. BCD dianggap sebagai kewajiban, karena masih ada 6 kombinasi yang tidak digunakan, tetapi tidak dapat digunakan untuk mewakili karakter lain. SBCDIC menggunakan kombinasi 6 bit, sehingga lebih banyak kombinasi yang dapat dihasilkan, sebanyak 64 kombinasi kode, yaitu 10 kode untuk digit angka, 26 kode untuk huruf alfabet dan sisa karakter khusus yang dipilih. Posisi bit dalam SBCDIC dibagi menjadi 2 zona, 2 bit pertama disebut posisi alfabet dan 4 bit berikutnya disebut posisi bit numerik.

3.4 Algoritma Pindah Ke Depan (MTF)

Algoritma ini akan mengambil string input S dengan N karakter S[0]... S[N-1] yang dipilih dari serangkaian karakter alfabet X. Untuk memberikan gambaran tentang teknik transformasi ini, contoh sederhana akan diambil menggunakan string S = 'abraca', N = 6, dan alfabet X = {'a' , 'b' , 'c', 'r'} yang akan dibagi menjadi 2 tahap, yaitu:

1. Urutkan rotasi

Matriks M membentuk orde N*N, yang elemen-elemennya adalah karakter dan garis-garisnya adalah rotasi dari setiap permutasi (pergeseran siklik) dari string S, yang diurutkan menurut abjad [leksikografis]. Salah satu baris dari matriks M adalah string asli S, sedangkan I adalah indeks yang menunjukkan urutan baris string asli dalam matriks M dengan penomoran dimulai dari nol.

Gambar 4. Pengurutan konsep matriks M

2. Ambil karakter terakhir dari rotasi

Berdasarkan konsep matriks ini, string L dapat dibentuk dari kolom terakhir dari matriks M, yaitu karakter L[0],...,L[N-1] sama dengan M [0, N-1] ,...,M[N-1, N-1] pada contoh sebelumnya, string L yang dihasilkan adalah 'caraab' dengan indeks I = 1. jadi dengan kata lain, hasil transformasi ini adalah L dan Saya pasang seperti yang terlihat pada gambar 3.6 di bawah ini.

Baris	L
0	aabrac
1	abraca
2	acaabr
3	bracaa
4	caabra
5	racaab

Gambar 5. hasil transformasi pada L dan I

Untuk memeriksa kerja Huffman Coding kita asumsikan bahwa kita memiliki file teks dan kita harus mengompresnya melalui Huffman Coding, Karakter dalam file ini memiliki frekuensi yang dapat ditampilkan sebagai berikut pada gambar 3.6:

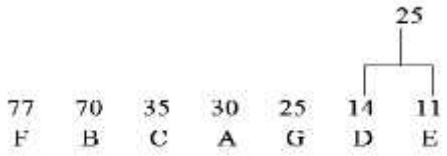
A:	30
B:	70
C:	35
D:	14
E:	11
F:	77
G:	25

Gambar 6. Karakter frekuensi dalam file

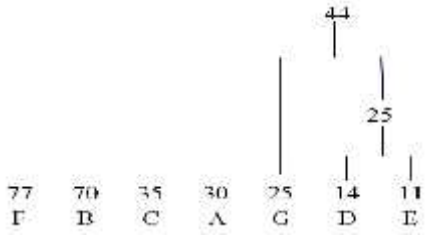
Kami memiliki karakter dari A ke G dan frekuensi yang sesuai. Langkah 1: Pada langkah pertama membangun Kode Huffman, Anda dapat mengurutkan karakter dari frekuensi kemunculan tertinggi hingga karakter terendah sebagai berikut:

77	70	35	30	25	14	11
F	B	C	A	G	D	E

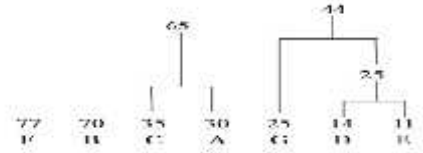
Langkah 2: Pada langkah kedua membangun kode Huffman, kita mengambil dua karakter yang paling jarang dan secara logis mengelompokkannya, dan kemudian frekuensinya ditambahkan. Dalam contoh kami, karakter D dan E telah dikelompokkan bersama dan kami memiliki frekuensi gabungan adalah 21:



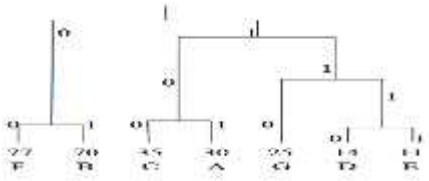
Ini memulai pembangunan struktur "pohon biner". Sekarang kita pilih lagi dua elemen dengan frekuensi terendah, dan frekuensi terendah adalah kombinasi D-E dan G. kita kelompokkan dan tambahkan frekuensinya. Ini adalah kombinasi baru dari frekuensi 44:



Anda dapat melanjutkan dengan cara yang sama untuk memilih dua elemen dengan frekuensi terendah dan mengelompokkannya bersama-sama dan kemudian menambahkan frekuensi, sampai kita mencapai semua elemen dan tetap hanya satu induk untuk semua node yang dikenal sebagai root. Pada hasil pengujian iterasi ketiga, elemen frekuensi terendah adalah C dan A:



Langkah 3: Pada langkah ketiga, beri label tepi dari setiap induk ke anak kirinya dengan angka 0 dan tepi ke anak kanan dengan angka 1. Simbol pengkodean untuk setiap huruf sumber adalah urutan label di sepanjang jalur dari akar ke simpul daun yang mewakili huruf itu. Sekarang pohon biner terakhir akan menjadi sebagai berikut:



Menelusuri pohon keputusan yang memberikan "kode Huffman", dengan kode terpendek yang ditetapkan untuk beberapa karakter dengan frekuensi yang lebih besar dapat ditunjukkan pada gambar 2.

F:	00
B:	01
C:	100
A:	101
G:	110
D:	1110
E:	1111

Gambar 7. Kode Huffman dengan kode terpendek

3.5 Algoritma Panjang Jalan

Algoritma Run-length dapat digunakan untuk mengompresi data analog yang berisi karakter berulang. Ketika karakter yang sama diterima dalam satu baris empat kali atau lebih (lebih dari tiga), algoritma memampatkan data dalam baris tiga karakter. Algoritma Run-Length paling efektif pada file grafik, yang biasanya berisi string karakter yang sama. Metode yang digunakan pada algoritma ini adalah dengan mencari karakter yang berulang lebih dari 3 kali dalam suatu file untuk diubah menjadi bit marker diikuti dengan bit yang memberikan informasi jumlah karakter yang berulang kemudian ditutup dengan karakter yang dikompresi, yang mana artinya penanda bit disini adalah rangkaian 8 bit yang membentuk sebuah karakter ASCII Jadi jika sebuah file berisi karakter berulang, seperti AAAAAAAAA atau dalam biner 01000001 8 kali, maka data tersebut dikompres menjadi 11111110 00001000 01000001. Dengan demikian kita dapat menyimpan sebagai sebanyak 5 byte.

3.6 Algoritma Pengkodean Aritmatika

Secara umum, algoritma kompresi data melakukan penggantian satu atau lebih simbol input dengan kode tertentu. Berbeda dengan itu, Pengodean Aritmatika dapat menggantikan string simbol input dengan angka floating point. Semakin lama dan semakin kompleks pesan yang dikodekan, semakin banyak bit yang dibutuhkan untuk tujuan itu. hasil dari pengkodean aritmatika ini adalah satu angka kurang dari 1 dan lebih besar dari atau sama dengan 0. Penomoran ini dapat didekodekan secara unik, untuk menghasilkan sekumpulan simbol yang digunakan dalam menghasilkan penomoran. Untuk dapat menghasilkan nilai keluaran, setiap simbol yang akan dikodekan diberi satu set nilai probabilitas.

3. HASIL DAN PEMBAHASAN

Metode dasar untuk kompresi data adalah Huffman pengkodean. Ini berfungsi untuk beberapa program populer digunakan di komputer pribadi. algoritma sederhana dan terutama digunakan untuk membuat pohon kode Huffman. Algoritma digunakan untuk membangun sebuah keputusan sebagai berikut:

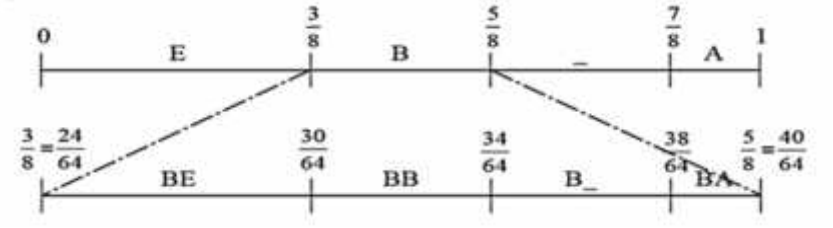
1. Kompresi
   diasumsikan bahwa kita memiliki string input dengan 8 karakter. Jika kita meletakkan pada array byte, kita mendapatkan array byte dengan ukuran 8 karakter. Satu karakter akan membutuhkan 8 bit jika karakter direpresentasikan dengan nilai ASCII. Satu set 8 bit dapat mewakili 28 karakter yang berbeda. Tetapi jika kita mempertimbangkan aplikasi, data teks sederhana mungkin disertakan hanya sekitar 26 karakter yang berbeda. Oleh karena itu perlu memiliki pengkodean 5 bit yang bisa menghasilkan 25 karakter yang berbeda untuk mewakili karakter yang lain. Untuk mengkonversi ke pengkodean 5-bit baru, ditetapkan nilai baru ke karakter alfabet seperti | a = 1 | b=2 | c=3 | d=4 | e=5 | f=6 | g=7 | h=8 |. Jika melihat lebih dekat pada array byte baru, itu akan terlihat seperti berikut (nilai karakter dalam representasi biner). 0000001 |00000010| 00000011| 000000100| 000000 101| 00000110| 00000111| 00001000| akan tetapi menggunakan 8 byte untuk menyimpan 8 karakter. Berikutnya langkah, kita akan memotong tiga bit dari posisi Bit ke-3 dari sisi kiri dan ekstrak 5 paling sedikit bit yang signifikan. Hasilnya akan ditampilkan sebagai berikut: |00001| 00010| 00011| 00100| 00101| 00110| 00111| 01000|. Sekarang kita telah mengurangi 8 byte menjadi 5 byte. Langkah selanjutnya menunjukkan bagaimana 5 byte ini dikonversi ke 8 byte dan kami mendapatkan yang asli informasi.
2. Dekompresi:
   Ketika array byte ditampilkan, setiap karakter harus direpresentasikan dalam bentuk biner. Kemudian semua 1 dan 0 harus diatur sebagai mereka nilai indeks dan semua data dibagi ke set lima bit. Setelah dilakukan pemisahan data, maka akan menjadi sebagai berikut:
   Kode |00001000| 10000110| 01000010| 10011000| 11101000| kemudian set ini diubah menjadi nilai desimal mewakili karakter yang kita telah dikompresi. Kode |00001 = 1(a) 000|10 = 2 (b) 00011 = 3(c) 0|0100 = 4 (d) 0010|1 = 5 (e) 00110 = 6 (f) 00|111 = 7 (g) 01000| = 8 (jam). Kemudian informasi akan ditampilkan dalam bentuk aslinya sebagai "ABCD EFGH".
   Encoding aritmatika adalah teknik kompresi yang paling kuat. Ini mengubah seluruh data input menjadi satu angka floating point. Angka floating point mirip dengan angka dengan titik desimal, seperti 4,5 bukannya 41/2. Namun, dalam pengkodean aritmatika kita tidak berurusan dengan angka desimal sehingga kita menyebutnya titik mengambang bukan titik desimal [4]. Mari kita ambil contoh kita memiliki string:

   BE_A_BEE

   Dan sekarang kami mengompresnya menggunakan pengkodean aritmatika.
   Langkah 1: pada langkah pertama yang kita lakukan adalah melihat hitungan frekuensi untuk huruf yang berbeda:

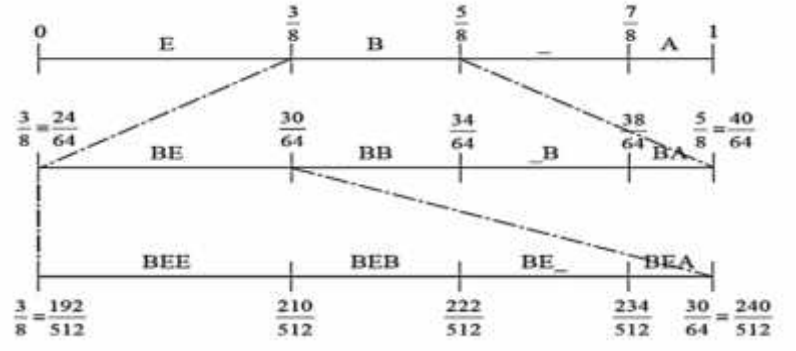
E	B	_	A
3	2	2	1

Langkah 2: Pada langkah kedua pengkodekan string dengan membagi interval [0, 1] dan mengalokasikan setiap huruf interval yang ukurannya tergantung pada seberapa sering itu dihitung dalam string. String kami dimulai dengan B‟, jadi kami mengambil interval B dan membaginya lagi dengan cara yang sama:

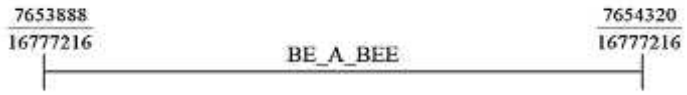


Batas antara BE‟ dan BB‟ adalah 3/8 dari jalan di sepanjang interval, yang panjangnya sendiri 2/3 dan dimulai pada 3/8. Jadi batasnya adalah 3/8 + (2/8) * (3/8) = 30/64. Demikian pula batas antara BB‟ dan B_‟ adalah 3/8+ (2/8) * (5/8) = 34/64, dan seterusnya.

Langkah 3: Pada langkah ketiga kita melihat huruf berikutnya sekarang E‟, jadi sekarang kita membagi interval E dengan cara yang sama. Kami melanjutkan melalui pesan .... Dan, melanjutkan dengan cara ini, kami akhirnya mendapatkan:



dan dilanjutkan dengan cara ini, kita peroleh:



Namun, tidak dapat mengirim nomor seperti 7654320/16777216 dengan mudah menggunakan komputer. Dalam notasi desimal, digit paling kanan di sebelah kiri titik desimal menunjukkan jumlah unit; yang di sebelah kirinya memberikan jumlah puluhan: yang berikutnya di sepanjang memberikan jumlah ratus, dan seterusnya. Jadi 7653888 = (7*106) + (6*105) + (5*104) + (3*103) + (8*102) + (8*10) + 8

Bilangan biner hampir persis sama, hanya kita berurusan dengan kekuatan 2 bukan kekuatan 10. Digit paling kanan dari bilangan biner adalah satuan (seperti sebelumnya) yang di sebelah kiri memberikan jumlah 2s, yang berikutnya nomor 4s, dan segera. Jadi 110100111 = (1*28) + (1*27) + (0*26) + (1*25) + (0*24) + (0*23) + (1*22) + (1*21) + 1 = 256 + 128 + 32 + 4 + 2 + 1 = 423

4. KESIMPULAN

Pengkodean aritmatika melewatkan gagasan untuk mengganti simbol input dengan kode tertentu. Algoritma ini menggantikan aliran simbol input dengan nomor output floating-point tunggal. Semakin banyak bit yang dibutuhkan dalam nomor keluaran, semakin rumit pesan yang diterima. Algoritma kompresi berbasis kamus menggunakan metode yang sangat berbeda untuk mengompresi data. Algoritma ini menggantikan string panjang variabel dari sebuah simbol menjadi sebuah token. Token adalah indeks dalam urutan kata dalam kamus. Jika token kecil dalam urutan kata, itu menggantikan prase dan kompresi terjadi, Pengkodean Huffman (1952)[10] memakai hampir semua karakteristik pengkodean Shannon-fano. Metode pengkodean Huffman dapat menghasilkan kompresi data yang efektif dengan mengurangi jumlah redundansi sinyal analog dalam simbol pengkodean. Telah terbukti bahwa pengkodean Huffman adalah metode panjang tetap yang paling efisien.

5. SARAN

Penulis berharap kedepanya kepada peniliti lainnya untuk dapat mengembangkan pnelitian ini yang tentunya tidak terlepas dari kekurangan baik dari unsur metode penelitiannya maupun judul penelitian untuk dapat melengkapi penelitian ini nantinya. Sehingga penelitian ini nantinya dapat digunakan untuk penelitian kedepannya

UCAPAN TERIMA KASIH

Ucapan Terima Kasih penulis sampaikan kepada rekan-rekan penulis dan peneliti atas masukannya dalam penelitian ini semoga kedepannya penelitian yang akan datang jauh lebih baik lagi.

DAFTAR PUSTAKA

[1] ZEWDIE WONDATIR, "*No Title No Title No Title*," *Angew. Chemie Int. Ed. 6(11), 951–952.*, vol. 10, no. April, pp. 64–72, 1967.
[2] R. Kaur, "*An Algorithm For Lossless Text Data Compression*," vol. 2, no. 7, 2013.
[3] A. Moffat, "*Huffman Coding*," *ACM Comput. Surv.*, vol. 52, no. 4, 2019, doi: 10.1145/3342555.
[4] S. Porwal, Y. Chaudhary, J. Joshi, and M. Jain, "*Data compression methodologies for lossless data and comparison between algorithms*," *Int. J. Eng. Sci. Innov. Technol.*, vol. 2, no. 2, pp. 142–147, 2013.
[5] R. Prabhakar and C. Rekha, "*VLSI Design and Implementation of Arithmetic and Logic Unit Using VHDL*," vol. 3, no. 10, pp. 62–67, 2013.
[6] L. A. Fitriya, T. W. Purboyo, and A. L. Prasasti, "*A review of data compression techniques*," *Int. J. Appl. Eng. Res.*, vol. 12, no. 19, pp. 8956–8963, 2017.
[7] D. J. Shoba and Dr.S.Sivakumar, "*A Study on Data Compression Using Huffman Coding Algorithms*," *Ijcst*, vol. 5, no. 1, pp. 58–63, 2017.
[8] G. G. Langdon, "*Introduction To Arithmetic Coding*.," *IBM J. Res. Dev.*, vol. 28, no. 2, pp. 135–149, 1984, doi: 10.1147/rd.282.0135.
[9] T. H. E. D. Age, "Shannon," *SpringerReference*, pp. 1–27, 2011, doi: 10.1007/springerreference_24599.
[10] *Introduction to*, vol. 12, no. 1s. 2015.
[11] S. N. Sulthana and M. Chandra, "*Image Compression with Adaptive Arithmetic Coding*," *Int. J. Comput. Appl.*, vol. 1, no. 18, pp. 35–37, 2010, doi: 10.5120/386-577.

[12] S. A. K. G. J. N. R. Shah, "*Advanced Data Compression Using J-bit Algorithm*," *Int. J. Sci. Res.*, vol. 4, no. 3, pp. 1366–1368, 2015, [Online]. Available: https://www.ijsr.net/archive/v4i3/SUB152218.pdf.
[13] "Data Compression 3 : *Arithmetic Coding the problem : encoding data succinctly*."
[14] R. S. Aarthi, D. Muralidharan, and P. Swaminathan, "Double compression of test data using Huffman code," *J. Theor. Appl. Inf. Technol.*, vol. 39, no. 2, pp. 104–113, 2012.
[15] S. T. Klein and D. Shapira, "On the randomness of compressed data," *Inf.*, vol. 11, no. 4, pp. 1–11, 2020, doi: 10.3390/info11040196.